BIBENCH: BENCHMARKING AND ANALYZING NETWORK BINARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural network binarization emerges as one of the most promising compression approaches with extraordinary computation and memory savings by minimizing the bit-width of weight and activation. However, despite being a generic technique, recent works reveal that applying binarization in a wide range of realistic scenarios involving diverse tasks, architectures, and hardware is not trivial. Moreover, common challenges, such as severe degradation in accuracy and limited efficiency gains, suggest that specific attributes of binarization are not thoroughly studied and adequately understood. To close this gap, we present BiBench, a rigorously designed benchmark with in-depth analysis for network binarization. We first carefully scrutinize the requirements of binarization in the actual production setting. We thus define the evaluation tracks and metrics for a fair and systematic investigation. We then perform a comprehensive evaluation with a rich collection of milestone binarization algorithms. Our benchmark results show binarization still faces severe accuracy challenges but diminishing improvements brought by newer state-of-the-art binarization algorithms, even at the expense of efficiency. Moreover, the actual deployment of certain binarization operations reveals a surprisingly large deviation from their theoretical consumption. Finally, we provide suggestions based on our benchmark results and analysis, devoted to establishing a paradigm for accurate and efficient binarization among existing techniques. We hope BiBench paves the way towards more extensive adoption of network binarization and serves as a foundation for future research.

1 INTRODUCTION

Since the rising of modern deep learning, the contradiction between ever-increasing model size and limited deployment resources has persisted. For this reason, the compression technologies are crucial to practical deep learning and have been widely studied, including model quantization (Gong et al., 2014; Wu et al., 2016; Vanhoucke et al., 2011; Gupta et al., 2015), network pruning (Han et al., 2015; 2016; He et al., 2017), knowledge distillation (Hinton et al., 2015; Xu et al., 2018; Chen et al., 2018; Yim et al., 2017; Zagoruyko & Komodakis, 2017), lightweight architecture design (Howard et al., 2017; Sandler et al., 2018; Zhang et al., 2018b; Ma et al., 2018), and low-rank decomposition (Denton et al., 2014; Lebedev et al., 2015; Jaderberg et al., 2014; Lebedev & Lempitsky, 2016).

As a compression approach that extremely reduces the bit-width, network binarization is seen as the most aggressive quantization technology. The binarized models leverage the most compact 1-bit parameters taking tiny storage and memory, and the efficient bitwise operations cost far less computation. Compared to other compression technologies like network pruning and architecture design, network binarization enjoys stronger topological generics since performing for parameters. Therefore, in academic research, network binarization is increasingly regarded as an independent compression technique and is widely studied instead of the 1-bit specialization of quantization. It is impressive that State-of-The-Art (SoTA) binarization algorithms are pushing binarized models to full-precision performance on some large-scale tasks (Deng et al., 2009; Liu et al., 2020).

However, existing network binarization technology is still far from practical. We point out that in binarization research, two worrisome trends are emerging for accuracy and efficiency:

Trend-1. **Accuracy comparison converging to limited scope.** In recent binarization research, image classification tasks (*e.g.*, ImageNet) are becoming standard options for accuracy comparisons.

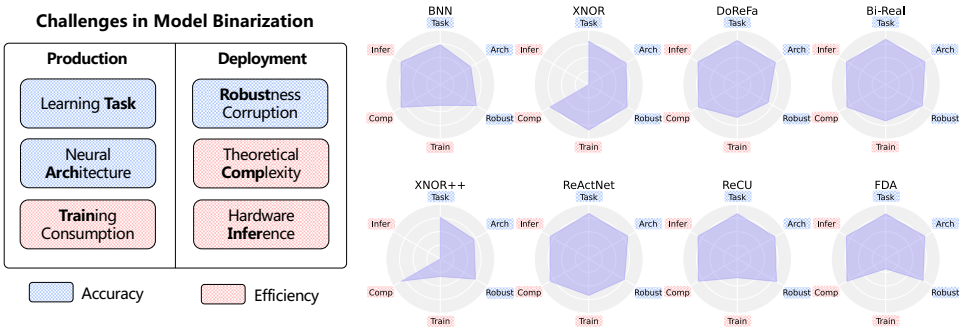


Figure 1: Evaluation tracks of BiBench. Our benchmark evaluates binarization algorithms on the most comprehensive evaluation tracks, including "Learning Task", "Neural Architecture", "Corruption Robustness", "Training Consumption", "Theoretical Complexity", and "Hardware Inference".

The common selection of evaluation tasks helps clearly compare the accuracy performance among different binarization algorithms, and also avoids unfair comparisons to a certain extent. However, as most mechanism studies for binarization algorithms are derived from the imaging modality tasks, the presented insights are rarely verified in a wider range of other modalities. The monotonic task also hinders the comprehensive evaluation from a architectural perspective, though binarization algorithms are usually architecture-general. Besides, data noise like corruption is a common problem on low-cost edge devices and is widely studied in compression (Lin et al., 2018; Rakin et al., 2021), whereas few advanced binarization algorithms consider the robustness of binarized models.

Trend-2. **Efficiency analysis remaining at the theoretical level.** network binarization is widely recognized for the significant memory and computational efficiency, *e.g.*, theoretical savings are up to 32× and 64× for convolutions, respectively (Rastegari et al., 2016; Bai et al., 2021). However, since the long-term absence of hardware deployment support, the models compressed by binarization algorithms are rarely evaluated for inference on real-world edge hardware, leaving their efficiency claims lacking practical evidence. In addition, the evaluation of the training process usually neglects in binarization research, which causes several negative phenomena in training, such as a large increase in training computation resources and time consumption, becoming sensitive to hyperparameters and requiring detailed tuning in optimization, *etc*.

In this paper, we present **BiBench**, a model **Bi**narization **Bench**mark to evaluate and analyze network binarization algorithms comprehensively from both accuracy and efficiency perspectives (Table 1). Based on BiBench, we benchmark 8 network binarization algorithms with 13 different neural architectures on 9 deep learning datasets (including 1 corruption benchmarks), 14 real-world hardware, and various hyper-parameter settings. We utilize about 4 GPU years of computation time to build our BiBench, devoting to pushing network binarization research to accurate and efficient. Furthermore, we analyze the benchmark results in depth to reveal several insights along comprehensive evaluation tracks and give suggestions for designing a practical binarization algorithm.

2 Background

2.1 Network Binarization

Binarization technology compresses weights $\boldsymbol{w}$ and activations $\boldsymbol{a}$ to 1-bit in computationally dense units (Courbariaux et al., 2016; Rastegari et al., 2016) and the computation can be expressed as

$$\boldsymbol{o} = \alpha \, \text{popcount}\left(\text{xnor}\left(\text{sign}(\boldsymbol{a}), \text{sign}(\boldsymbol{w})\right)\right), \tag{1}$$

where $\boldsymbol{o}$ and α denotes the outputs and the optional scaling factor. While enjoying the extreme compression and acceleration, severely limited representation causes the degradation of binarized models, thereby various algorithms constantly emerge to improve accuracy (Yuan & Agaian, 2021).

The vast majority of binarization algorithms are designed to improve the binarized operators. As shown in Eq. (1), the fundamental difference between binarized and full-precision models is the former applies binarized operators. The operators directly affect the optimization properties of binarized models and also determine their efficiency when deployment (Alizadeh et al., 2019; Geiger & Team, 2020). In addition, the improvement of operators is widely applicable for various architectures and tasks (Wang et al., 2020; Qin et al., 2022; Zhao et al., 2022), which benefits algorithms

Table 1: Comparison between BiBench and previous binarization works along evaluation tracks.

Algorithm	Technique			Accurate Binarization			Efficient Binarization		
	s	τ	g	#Task	#Arch.	Corruption Robustness	Training Consumption	Theoretical Complexity	Hardware Inference
BNN (Courbariaux et al., 2016)	×	×	√	3	3	*	√	√	√
XNOR (Rastegari et al., 2016)	√	×	×	2	3	*	√	√	√
DoReFa (Zhou et al., 2016)	√	×	×	2	2	*	×	√	×
Bi-Real (Liu et al., 2018)	×	×	√	1	2	×	×	√	×
XNOR++ (Bulat et al., 2019)	√	×	×	1	2	×	×	×	×
ReActNet (Liu et al., 2020)	×	√	×	1	2	×	×	√	×
ReCU (Xu et al., 2021b)	×	√	√	2	4	×	×	×	×
FDA (Xu et al., 2021a)	×	×	√	1	6	×	×	×	×
*Our Benchmark (**BiBench**)*	√	√	√	**9**	**13**	√	√	√	√

[1] "√" and "×" indicates the track is considered when the binarization algorithm was proposed, while "*" indicates only being studied in other related studies. "s", "τ", or "g" indicates "scaling factor", "parameter redistribution", or "gradient approximation" techniques proposed in this work, respectively.

to fully exert the generic of binarization technology. We thereby consider 8 generic binarization algorithms focusing on operator improvement in our BiBench (Courbariaux et al., 2016; Rastegari et al., 2016; Zhou et al., 2016; Liu et al., 2018; Bulat et al., 2019; Liu et al., 2020; Xu et al., 2021b;a). These binarization algorithms propose different techniques, which improve the binarization performance mainly by techniques with scaling factors, parameter redistribution, and gradient approximation, respectively. Note that the techniques requiring specified local structures or training pipelines in algorithms are excluded to cope with the diverse architectures and training processes in the evaluation. For a detailed description of algorithms, please see Appendix. A.

2.2 Challenges for Binarization

Since about 2015, network binarization has made it of great interest in different applications, including but not limited to vision, language understanding, *etc*. However, in practice, there are still various challenges in the practical production and deployment process of binarization.

The goal in production is to train accurate binarized models with controllable resources. Recent work has revealed the capability of binarization algorithms evaluated on image classification is not always appliable to new learning tasks and neural architectures (Qin et al., 2020; Wang et al., 2020; Qin et al., 2021; Liu et al., 2022). And to achieve higher accuracy, some binarization algorithms require several times of training resources compared with the training of full-precision models. When deployed to edge devices, binarized models are expected to be hardware-friendly and robust. Unlike multi-bit (2-8 bit) quantization, the deployment of binarization is not supported by most mainstream hardware inference libraries (NVIDIA, 2022; HUAWEI, 2022; Qualcomm, 2022), which also leads to the fact that the theoretical efficiency of existing algorithms is rarely verified on actual hardware. Moreover, since the data collected by low-cost devices in edge scenarios is not always clean and high-quality, the robustness of quantized and binarized models starts to be studied (Lin et al., 2018; Ye et al., 2019; Cygert & Czyżewski, 2021). However, the robustness is still rarely considered in the designing of binarization algorithms.

3 BiBench: Tracks and Metrics for Binarization

In this section, we present BiBench towards accurate and efficient network binarization. As Figure 1 shows, we build 6 evaluation tracks and corresponding evaluation metrics upon the practical challenges in the production and deployment process of network binarization. In all evaluation tracks, higher metrics mean better performance.

3.1 Towards Accurate Binarization

The evaluation tracks we focus on for accurate network binarization are "Learning Task", "Neural Architecture", and "Corruption Robustness".

① **Learning Task**. To comprehensively evaluate network binarization algorithms, we selected 9 classes of deep learning tasks with 4 classes of different data modalities. For the widely-

evaluated 2D visual modality, in addition to the widely used image classification tasks on CIFAR-10 (Krizhevsky et al., 2014) and ImageNet (Krizhevsky et al., 2012), object detection tasks on PASCAL VOC (Hoiem et al., 2009) and COCO (Lin et al., 2014) are also considered across all algorithms. For the 3D visual modality, we evaluate the algorithm on ModelNet40 classification (Wu et al., 2015) and ShapeNet segmentation (Chang et al., 2015) tasks of 3D point clouds. For textual modality, the natural language understanding tasks in GLUE benchmark (Wang et al., 2018) are applied for evaluation. For speech modality, we evaluate binarization algorithms on Speech Commands KWS task (Warden, 2018). The details of these tasks and datasets are presented in Appendix. B.

Then we build an evaluation metric for this track. For a certain binarization algorithm, we take the accuracy of full-precision models as baselines and calculate the mean relative accuracy for all architectures on each task. Then we calculate the Overall Metric (OM) of the task track as the quadratic mean of all tasks (Curtis & Marshall, 2000). The equation of evaluation metric is

$$\text{OM}_{\text{task}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}^2\left(\frac{\boldsymbol{A}^{bi}_{\text{task}_i}}{\boldsymbol{A}_{\text{task}_i}}\right)}, \tag{2}$$

where $\boldsymbol{A}^{bi}_{\text{task}_i}$ and $\boldsymbol{A}_{\text{task}_i}$ denote the accuracy results of the binarized and full-precision models on i-th task, N is the number of tasks, and $\mathbb{E}(\cdot)$ denote taking the mean value. The quadratic mean form is uniformly applied in BiBench to unify all metrics.

② **Neural Architecture**. We choose diverse neural architectures with mainstream CNN-based, transformer-based, and MLP-based backbones, to evaluate the generalizability of binarization algorithms for architectures. We adopt standard ResNet-18/20/34 (He et al., 2016) and VGG (Simonyan & Zisserman, 2015) in the binarization evaluation of CNN architectures, and the Faster-RCNN (Ren et al., 2015) and SSD300 (Liu et al., 2016) frameworks with ResNet-18 and VGG backbones are also applied. For transformer-based architectures with attention mechanism, we binarized BERT-Tiny4/Tiny6/Base (Kenton & Toutanova, 2019) with different model sizes. And PointNet (Vanilla), PointNet (Qi et al., 2017), FSMN (Zhang et al., 2015), and Deep-FSMN (Zhang et al., 2018a) are seen as MLP-based architectures for their linear unit composition. The details of these architectures are presented in Appendix. C.

Similar to the overall metric in the task track, we build that for the architecture track as follows:

$$\text{OM}_{\text{arch}} = \sqrt{\frac{1}{3}\left(\mathbb{E}^2\left(\frac{\boldsymbol{A}^{bi}_{\text{CNN}}}{\boldsymbol{A}_{\text{CNN}}}\right) + \mathbb{E}^2\left(\frac{\boldsymbol{A}^{bi}_{\text{Transformer}}}{\boldsymbol{A}_{\text{Transformer}}}\right) + \mathbb{E}^2\left(\frac{\boldsymbol{A}^{bi}_{\text{MLP}}}{\boldsymbol{A}_{\text{MLP}}}\right)\right)}. \tag{3}$$

③ **Corruption Robustness**. The corruption robustness of binarization on deployment is important to deal with bad cases like perceptual device damage, which is a common problem with low-cost equipment in real-world deployments. We consider the robustness of binarized models to corruption of 2D visual data, and evaluate algorithms on the CIFAR10-C (Hendrycks & Dietterich, 2018) benchmark.

We focus on evaluating how well a binarized model could behave towards the corrupted data compared with the normal data, and thus use the corruption generalization gap (Zhang et al., 2022):

$$\boldsymbol{G}_{\text{task}_i} = \boldsymbol{A}^{\text{norm}}_{\text{task}_i} - \boldsymbol{A}^{\text{corr}}_{\text{task}_i}, \tag{4}$$

where $\boldsymbol{A}^{\text{corr}}_{\text{task}_i}$ and $\boldsymbol{A}^{\text{norm}}_{\text{task}_i}$ denote the accuracy results of all architectures on i-th corruption task and corresponding normal task, respectively. And the overall metric on this track is calculated by

$$\text{OM}_{\text{robust}} = \sqrt{\frac{1}{C}\sum_{i=1}^{C}\mathbb{E}^2\left(\frac{\boldsymbol{G}_{\text{task}_i}}{\boldsymbol{G}^{bi}_{\text{task}_i}}\right)}. \tag{5}$$

3.2 Towards Efficient Binarization

As for the efficiency of binarization, "Training Consumption" in production and "Theoretical Complexity" and "Hardware Inference" in deployment are considered.

④ **Training Consumption**. We consider the training resource occupation and hyper-parameter sensitivity of binarization algorithms, which affect the consumption of single training and complete

tuning process, respectively. On a same task, various training settings of hyper-parameters are divided, including different learning rates, learning rate schedulers, optimizers, and random seeds, to determine how easy the algorithm is to tune the model to an optimal state. And we align the epoch numbers in binarization with full-precision counterparts and compare their consumption and time.

The evaluation metric for this track is related to the training time and hyper-parameter sensitivity. For a specific binarization algorithm, we have

$$\mathrm{OM}_{\text{train}} = \sqrt{\frac{1}{2}\left(\mathbb{E}^2\left(\frac{\boldsymbol{T}_{\text{train}}}{\boldsymbol{T}^{bi}_{\text{train}}}\right) + \mathbb{E}^2\left(\frac{\mathrm{std}(\boldsymbol{A}_{\text{hyper}})}{\mathrm{std}(\boldsymbol{A}^{bi}_{\text{hyper}})}\right)\right)}, \tag{6}$$

where $\boldsymbol{T}_{\text{train}}$ denotes the set of time in once training, $\boldsymbol{A}_{\text{hyper}}$ is the set of results with different hyper-parameter settings, and $\mathrm{std}(\cdot)$ is taking standard deviation values.

⑤ **Theoretical Complexity**. When evaluating theoretical complexity, we calculate the compression and speedup ratio between before and after binarization on architectures such as ResNet18.

The evaluation metric is calculated by the saving of model size (MB) and computational floating-point operations (FLOPs) at inference. For binarized parameters, the storage occupation of binarized parameters is 1/32 as their 32-bit floating-point counterparts (Rastegari et al., 2016). For binarized operations, the multiplication between a 1-bit number (weight) and a 1-bit number (activation) approximately takes 1*1/64 FLOPs for a CPU with the instruction size of 64 bits (Zhou et al., 2016; Liu et al., 2018; Li et al., 2019). The compression ratio $\boldsymbol{r}_c$ and speedup ratio $\boldsymbol{r}_s$ are

$$\boldsymbol{r}_c = \frac{|\boldsymbol{M}|_{\ell 0}}{\frac{1}{32}\left(|\boldsymbol{M}|_{\ell 0} - |\hat{\boldsymbol{M}}|_{\ell 0}\right) + |\hat{\boldsymbol{M}}|_{\ell 0}}, \quad \boldsymbol{r}_s = \frac{\mathrm{FLOPs}_{\boldsymbol{M}}}{\frac{1}{64}\left(\mathrm{FLOPs}_{\boldsymbol{M}} - \mathrm{FLOPs}_{\hat{\boldsymbol{M}}}\right) + \mathrm{FLOPs}_{\hat{\boldsymbol{M}}}}, \tag{7}$$

where $\boldsymbol{M}$ and $\hat{\boldsymbol{M}}$ are the parameters of the original full-precision model and the 32-bit part of binarized model, respectively, and $\mathrm{FLOPs}_{\boldsymbol{M}}$ and $\mathrm{FLOPs}_{\hat{\boldsymbol{M}}}$ denote the computation related to these parameters, respectively. And the overall metric for theoretical complexity is

$$\mathrm{OM}_{\text{comp}} = \sqrt{\frac{1}{2}\left(\mathbb{E}^2(\boldsymbol{r}_c) + \mathbb{E}^2(\boldsymbol{r}_s)\right)}. \tag{8}$$

⑥ **Hardware Inference**. Since the extremely limited support of binarization in hardware deployment, just 2 inference libraries, Larq's Compute Engine (Geiger & Team, 2020) and JD's daBNN (Zhang et al., 2019), can deploy and evaluate the binarized models on ARM hardware in practice. Regarding hardware, we mainly focus on ARM CPU inference as this is the main hardware type for edge scenarios, including HUAWEI Kirin, Qualcomm Snapdragon, Apple M1, MediaTek Dimensity, and Raspberry Pi. We put the details of the hardware in Appendix. D.

And for a certain binarization algorithm, we regard the saving of storage and inference time under different inference libraries and hardware as specific evaluation metrics:

$$\mathrm{OM}_{\text{infer}} = \sqrt{\frac{1}{2}\left(\mathbb{E}^2\left(\frac{\boldsymbol{T}_{\text{infer}}}{\boldsymbol{T}^{bi}_{\text{infer}}}\right) + \mathbb{E}^2\left(\frac{\boldsymbol{S}_{\text{infer}}}{\boldsymbol{S}^{bi}_{\text{infer}}}\right)\right)}, \tag{9}$$

where $\boldsymbol{T}_{\text{infer}}$ and $\boldsymbol{S}_{\text{infer}}$ denote the inference time and storage on different devices, respectively.

4 BiBench Implementation

In this section, we show the implementation details, training and inference pipelines, and main results of our binarization benchmark.

Implementation details. We implement BiBench with PyTorch (Paszke et al., 2019) packages. The definition of the binarized operator part is completely independent. The operator defined by each binarization algorithm is abstracted into a single file, and the corresponding operator in the original model can be flexibly replaced in the evaluation of different tasks and architectures. When deployment, the well-trained binarized model is exported in the ONNX form accepted by the inference libraries in BiBench (if the binarization algorithm is applicable).

Table 2: Accuracy benchmark for model binarization.

Track	Metric	Binarization Algorithm							
		BNN	XNOR	DoReFa	Bi-Real	XNOR++	ReActNet	ReCU	FDA
Learning Task (%)	CIFAR10	94.54	94.73	95.03	95.61	94.52	95.92	96.72	94.66
	ImageNet	75.81	77.24	76.61	78.38	75.01	78.64	77.98	78.15
	VOC07	76.97	74.61	76.35	80.07	74.41	81.38	81.65	79.02
	COCO17	77.94	75.37	78.31	81.62	79.41	83.82	85.66	82.35
	ModelNet40	54.19	93.86	93.74	93.23	85.20	92.41	95.07	94.38
	ShapeNet	48.96	73.62	70.79	68.13	41.16	68.51	40.65	71.16
	GLUE	49.75	59.63	66.60	69.42	49.33	67.64	50.66	70.61
	SpeechCom.	75.03	76.93	76.64	82.42	68.65	81.86	76.98	77.90
	$\mathbf{OM_{task}}$	70.82	78.97	79.82	81.63	72.89	81.81	77.96	81.49
Neural Architecture (%)	CNNs	72.90	83.74	83.86	85.02	78.95	86.20	83.50	86.34
	Transformers	49.75	59.63	66.60	69.42	49.33	67.64	50.66	70.61
	MLPs	64.61	85.40	85.19	87.83	76.92	87.13	86.02	86.14
	$\mathbf{OM_{arch}}$	63.15	77.16	79.01	81.16	69.72	80.82	75.14	81.36
Robustness Corruption (%)	CIFAR10-C	95.26	100.97	81.43	96.56	92.69	94.01	103.29	98.35
	$\mathbf{OM_{corr}}$	95.26	100.97	81.43	96.56	92.69	94.01	103.29	98.35

Training and inference pipelines. *Hype-parameters*: We train the $1\times$ number of training epochs for binarized models compared to their full-precision counterparts. Inspired by results in Section 5.2.1, Adam optimizer is applied to all binarized models to make them converge well, the initial learning rates are set to $0.1\times$ (or $1e-3$ as default), and the learning rate scheduler is CosineAnnealingLR. *Architecture*: In BiBench, we completely follow the original architectures of full-precision models, and binarizing their convolution, linear, and multiplication units with the binarization algorithms. Hardtanh is uniformly used as the activation function to avoid the all-one feature. *Pretrains*: We adopt finetuning for all algorithms, each binarized model is initialized by the same pre-trained model for a same architecture and task, to eliminate the inconsistency at initialization.

5 BiBench Evaluation

The accuracy results of binarization algorithms are shown in Table 2, and the efficiency results are shown in Table 3 and Figure 2.

5.1 Accuracy Analysis for Binarization

For the accuracy results of network binarization, we present the results in Table 2 by each accuracy tracks and report the associated metrics.

5.1.1 Learning Task Track

We present the evaluation results of binarization algorithms on various learning tasks. Besides the overall metrics $\mathrm{OM}_{\mathrm{task}}$, we also reported the relative accuracy results of these tasks.

Accuracy improvement is still the most rigorous challenge in binarization algorithm research so far. With fully unified training pipelines and neural architectures, we find the performance of binarization is still limited on most learning tasks. The results on large-scale ImageNet and COCO is still usually less than 80% of their full-precision counterparts. Moreover, the marginal effects of improvement are starting to appear in advanced binarization algorithms, *e.g.*, the the SoTA ReCU is less 3% higher than vanilla BNN on the ImageNet.

Accuracy of binarization algorithms differs a lot under different data modalities. The binarized models suffer a huge accuracy drop on text understanding GLUE benchmark, while it is almost able to approach full-precision performance on the ModelNet40 point cloud classification task. This suggests that the transfer of binarization studies' insights across different tasks is non-trivial.

As for the overall performance, both ReCU and ReActeNet show high accuracy across different tasks. Surprisingly, although ReCU wins the championship on the most 4 individual tasks, ReActNet finally stands out in the overall metric comparison. Both of them use reparameterization in the forward propagation and gradient approximation in the backward propagation.

Table 3: Efficiency benchmark for model binarization.

Track	Metric	Binarization Algorithm							
		BNN	XNOR	DoReFa	Bi-Real	XNOR++	ReActNet	ReCU	FDA
Training Consumption (%)	Sensitivity	27.28	175.53	113.40	144.59	28.66	146.06	53.33	36.62
	Time Cost	82.19	71.43	76.92	45.80	68.18	45.45	58.25	20.62
	$\mathbf{OM_{train}}$	61.23	134.00	96.89	107.25	52.30	108.16	55.84	29.72
Theoretical Complexity (×)	Speedup	12.60	12.26	12.37	12.37	12.26	12.26	12.37	12.37
	Compression	13.27	13.20	13.20	13.20	13.16	13.20	13.20	13.20
	$\mathbf{OM_{comp}}$	12.94	12.74	12.79	12.79	12.71	12.74	12.79	12.79
Hardware Deployment (×)	Speedup	5.45	False	5.45	5.45	False	4.89	5.45	5.45
	Compression	15.62	False	15.62	15.62	False	15.52	15.62	15.62
	$\mathbf{OM_{infer}}$	11.70	False	11.70	11.70	False	11.51	11.70	11.70

5.1.2 Neural Architecture Track

Binarization exhibits a clear advantage on CNN-based and MLP-based architectures over transformer-based ones. Since being widely studied, the advanced binarization algorithms usually achieve about 78% 86% metric in CNN. And the binarized models with MLP architectures even approach the full-precision level performance (*e.g.*, Bi-Real Net 87.83%). In stark contrast, the transformer-based one suffers from extremely significant performance degradation when binarization and none of the algorithms achieves an overall metric higher than 70%. The results that compared to CNNs and MLPs, transformer-based architectures are driven by a unique attention mechanism, which requires specifically studied in binarization research.

The overall winner on the architecture track is the FDA algorithm. In this track, FDA has achieved the best results in both CNN and Transformer. The evaluation of these two tracks proves that these types of binarization algorithms like FDA and ReActNet have a certain sense of stability advantage, and both of them apply statistical channel-wise scaling factor and custom gradient approximation.

5.1.3 Corruption Robustness Track

The binarized model can approach full-precision level robustness for corruption. Surprisingly, the binarized model exhibits robustness close to the full-precision model in evaluation on corrupted data, despite severely limited representation capability and a significant drop in accuracy. Results on the CIFAR10-C dataset show that the binarized model has performance close to the full-precision model in the common 2D image corruption, and ReCU and XNOR-Net even outperform their full-precision counterparts. This fact means that if the requirements for the model's corruption robustness are not the same, the binarized model requires almost no additional training or supervision for robustness. At the same time, various binarization algorithms show close performance, so this property can be regarded as a general property of the binarized model, not just for a specific algorithm.

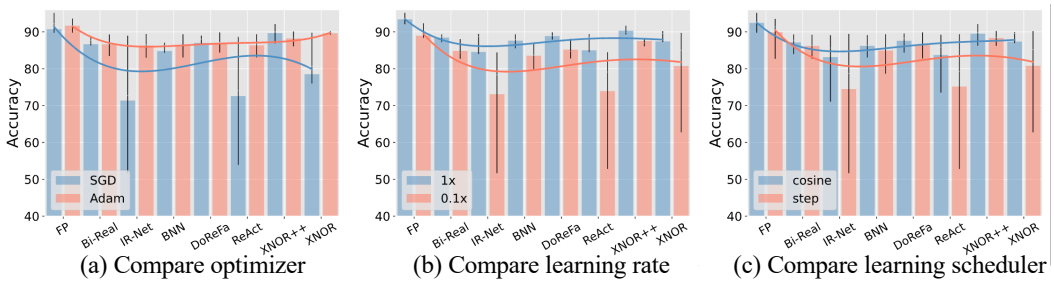
5.2 Efficiency Analysis for Binarization

For the efficiency results, the metrics of training consumption, theoretical complexity, and hardware inference tracks are presented.

5.2.1 Training Consumption Track

We comprehensively investigate the training cost of the binarization algorithm on ResNet18 of CIFAR10. Furthermore, we present the results of sensitivity and training time for different binarization algorithms in Table 3 and Figure 3.

"Binarization≠Sensitivity": existing techniques can stable the training of binarization. An existing common intuition is that the training of binarized models was more sensitive to the training settings than full-precision models, due to the representational limitations and gradient approximation errors brought about by the high degree of discretization. However, we find that the hyperparameter sensitivities of existing binarization algorithms are polarized when using the exact same training process. Some of them are even more stable for hyperparameters than full-precision training, while the other part fluctuates hugely. The reason for this problem is the difference in the techniques applied by the binarized operators of these algorithms. The training-stable binarization algorithms often have the following commonalities: (1) *Channel-wise floating-point scaling factors*, which are based on learning or statistics; (2) *Soft gradient approximation* to reduce gradient error.



(a) Compare optimizer (b) Compare learning rate (c) Compare learning scheduler

Figure 2: Comparisons of accuracy under different training settings.

These hyperparameter-stable binarization algorithms may not certainly outperform other algorithms but are able to simplify the tuning process in production and obtain reliable accuracy in one training.

The preference for hyperparameter settings is evident for the training of binarization. From the statistical results in Figure 2, Adam optimizer, $1\times$ (same to full-precision model) learning rate and CosineAnnealingLR scheduler are more stable than other settings. Inspired by this, we adopt this setting in the evaluation of binarization as part of the standard training pipelines.

The soft gradient approximation in binarization brings a major training time increase. Comparing the time consumed by each binarization algorithm, we found that the training time of algorithms using custom gradient approximation techniques such as Bi-Real and ReActNet increased significantly, and the FDA index was even as low as 20.62%, which means that the training time he spent was close to $5\times$ the full-precision model training.

5.2.2 Theoretical Complexity Track

There is no obvious difference in the theoretical complexity among binarization algorithms. The main source of the difference in compression rate is the definition of the static scaling factor of each model. For example, BNN does not have a scale factor. For theoretical acceleration, the main difference comes from two aspects. First, the compression of the static scaling factor also improves the theoretical speedup. Second, real-time re-scaling and mean-shifting for activation bring additional theoretical calculations, such as ReActNet, which harms $0.11\times$ speedup. In general, the theoretical complexity of each method is similar, and the overall metrics are in the range of $[12.71, 12.94]$. These results theoretically show that the various binarization algorithms have similar inference efficiency.

5.2.3 Hardware Inference Track

Compared to other tracks, the hardware inference track brings us some insights for binarization in real-world deployment.

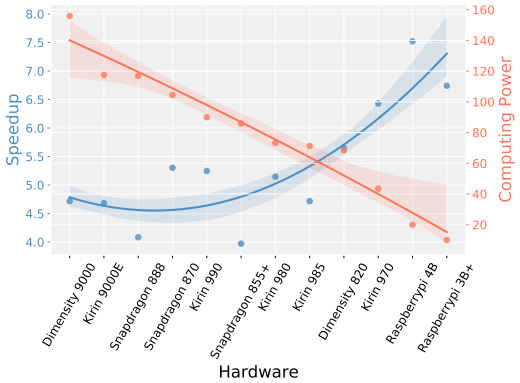


Figure 3: The lower the computing power of the hardware chip, the higher inference speedup is achieved by deployed binarized models.

Extremely limited inference libraries leads to an almost fixed paradigm of binarization deployment. After investigating existing open-source inference libraries, we find that there are very few inference libraries that support the real deployment of binarization algorithms on hardware. And there are only Larq (Geiger & Team, 2020) and daBNN (Zhang et al., 2019) with a complete deployment pipeline and mainly support deployment on ARM devices. We first evaluate the deployment capability of the two inference libraries in Table 4. Both two inference libraries support a channel-wise scaling factor in floating-point form and force it to be fused into the BN layer (which means that every convolution of the binarized model must be followed by BN), and neither supports dynamic activation of activations Statistics and re-scaling in inference. The only difference is that Larq further supports mean-shifting activation with a fixed bias. Constrained by the inference libraries, the practical deployment of binarization algorithms is limited. The scale factor shape of XNOR++ led to its failed deployment, and XNOR also failed because of the activation re-scaling technique. The vast majority of binarization methods have almost identical inference performance, and the mean-shifting operation of ReActNet on activation slightly affects the effi-

Table 4: Deployment capability of different inference libraries on real-hardware.

Infer. Lib.	Provider	s Granularity	s Form	Flod BN	Act. Re-scaling	Act. Mean-shifting
Larq	Larq	Channel-wise	FP32	$\surd$	$\times$	$\surd$
daBNN	JD	Channel-wise	FP32	$\surd$	$\times$	$\times$
Algorithm	**Deployable**	**s Granularity**	**s Form**	**Flod BN**	**Act. Re-scaling**	**Act. Mean-shifting**
BNN	$\surd$	N/A	N/A	N/A	$\times$	$\times$
XNOR	$\times$	Channel-wise	FP32	$\surd$	$\surd$	$\times$
DoReFa	$\surd$	Channel-wise	FP32	$\surd$	$\times$	$\times$
Bi-Real	$\surd$	Channel-wise	FP32	$\surd$	$\times$	$\times$
XNOR++	$\times$	Spatial-wise	FP32	$\times$	$\times$	$\times$
ReActNet	$\surd$	Channel-wise	FP32	$\surd$	$\times$	$\surd$
ReCU	$\surd$	Channel-wise	FP32	$\surd$	$\times$	$\times$
FDA	$\surd$	Channel-wise	FP32	$\surd$	$\times$	$\times$

ciency, *i.e.*, binarized models must satisfy fixed deployment paradigms and have almost identical performance.

Born for the edge: the lower computing power of chip, the higher binarization speedup achieved.. After deploying and evaluating binarized models across dozen chips, we compare the average speedup of the binarization algorithm on each chip. A counter-intuitive finding is that the higher the performance, the more worse the binarization speedup on the chip. Through further observation, we found that the gap is mainly due to the use of higher thread calculations. Higher performance chips have more acceleration brought by multi-threading when running floating-point models, and thus the speedup of binarized models is relatively slow. This means that the scenarios where binarization technology really comes into play are on edge chips with low performance and cost, and binarization extreme compression and acceleration is precisely what makes this possible.

5.3 Suggestions for Binarization Algorithm

Based on the above evaluation and analysis, we attempt to summarize a paradigm for accurate and efficient binarization among existing binarization techniques: (1) **Soft quantization approximation** is an undisputed must-have technique. This technique does not affect hardware inference efficiency and is adopted by all winning binarization algorithms on accuracy tracks, including Bi-Real, ReActNet, and ReCU. (2) **Channel-wise scaling factors** are the only option available for practical binarization. The results on the task and architecture tracks demonstrate the advantage of floating-point scaling factors, and analysis on efficiency tracks further limits it to the channel-wise form. (3) **Mean-shifting the input with a fixed bias is an optional helpful operation.** Our results show that this technique in ReActNet is effective in improving accuracy and consumes almost no extra computation, while it is not supported by all inference libraries.

We have to stress that although the evaluation on 6 tracks leads us to some ground rules for network binarization, none of the binarization techniques or algorithms can work well across all scenarios so far. In the future, binarization research should focus on breaking the mutual restrictions between production and deployment, the binarization algorithms should follow the deployable orientation while the inference libraries should support binarized operators with higher accuracy.

6 Discussion

In this paper, we present BiBench, a versatile and comprehensive benchmark that delves into the most fundamental questions of model binarization. BiBench covers 8 network binarization algorithms, 13 different neural architectures, 9 deep learning datasets (including 1 corruption benchmarks), 14 real-world hardware, and various hyper-parameter settings. Moreover, BiBench proposes evaluation tracks specifically designed to gauge critical aspects such as accuracy under diverse circumstances, and efficiency especially when deployed on actual hardwares. More importantly, by collating experiment results and analysis, BiBench hopes to establish an empirically optimized paradigm with several key considerations for designing accurate and efficient binarization methods. We hope BiBench could facilitate a fair comparison of algorithms through a systematic investigation with metrics that reflect the realistic requirements, and serve as a foundation for applying model binarization in a broader and more practical context.

REFERENCES

Milad Alizadeh, Javier Fernández-Marqués, Nicholas D. Lane, and Yarin Gal. A systematic study of binary neural networks' optimisation. In *ICLR*, 2019.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael R Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. In *ACL/IJCNLP (1)*, 2021.

Adrian Bulat, Georgios Tzimiropoulos, and Samsung AI Center. Xnor-net++: Improved binary neural networks. In *British Machine Vision Conference*, 2019.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Yuntao Chen, Zhaoxiang Zhang, and Naiyan Wang. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. *national conference on artificial intelligence*, pp. 2852–2859, 2018.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Robert O Curtis and David D Marshall. Why quadratic mean diameter? *Western Journal of Applied Forestry*, 15(3):137–139, 2000.

Sebastian Cygert and Andrzej Czyżewski. Robustness in compressed neural networks for object detection. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li, and Fei Fei Li. Imagenet: a large-scale hierarchical image database. In *IEEE CVPR*, 2009.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 1269–1277. Curran Associates, Inc., 2014.

Lukas Geiger and Plumerai Team. Larq: An open-source library for training binarized neural networks. *Journal of Open Source Software*, 5(45):1746, 2020.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv: Computer Vision and Pattern Recognition*, 2014.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. *ICML*, pp. 1737–1746, 2015.

Song Han, Jeff Pool, John Tran, and William 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. In *ICLR*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE CVPR*, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE ICCV*, 2017.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

Geoffrey E Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv: Machine Learning*, 2015.

Hisilicon. Kirin chipsets, 2022. URL https://www.hisilicon.com/cn/products/Kirin.

Derek Hoiem, Santosh K Divvala, and James H Hays. Pascal voc 2008 challenge. *World Literature Today*, 24, 2009.

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 preprint arXiv:1704.04861*, 2017.

HUAWEI. Acl: Quantization factor file, 2022. URL https://support.huaweicloud.com/intl/en-us/ti-mc-A200_3000/altasmodelling_16_043.html.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 141(5):1097–1105, 2012.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *IEEE CVPR*, June 2016.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V Oseledets, and Victor S Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *ICLR*, 2015.

Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2019.

Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. In *International Conference on Learning Representations*, 2018.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Zechun Liu, Baoyuan Wu, Wenhan 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 *Proceedings of the European conference on computer vision (ECCV)*, pp. 722–737, 2018.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *European conference on computer vision*, pp. 143–159. Springer, 2020.

Zechun Liu, Barlas Oguz, Aasish Pappu, Lin Xiao, Scott Yih, Meng Li, Raghuraman Krishnamoorthi, and Yashar Mehdad. Bit: Robustly binarized multi-distilled transformer. In *Neural Information Processing System*, 2022.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pp. 116–131, 2018.

MediaTek. Mediatek 5g, 2022. URL https://i.mediatek.com/mediatek-5g.

NVIDIA. Tensorrt: A c++ library for high performance inference on nvidia gpus and deep learning accelerators, 2022. URL https://github.com/NVIDIA/TensorRT.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Haotong Qin, Zhongang Cai, Mingyuan Zhang, Yifu Ding, Haiyu Zhao, Shuai Yi, Xianglong Liu, and Hao Su. Bipointnet: Binary neural network for point clouds. In *International Conference on Learning Representations*, 2020.

Haotong Qin, Yifu Ding, Mingyuan Zhang, YAN Qinghua, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. Bibert: Accurate fully binarized bert. In *International Conference on Learning Representations*, 2021.

Haotong Qin, Xudong Ma, Yifu Ding, Xiaoyang Li, Yang Zhang, Yao Tian, Zejun Ma, Jie Luo, and Xianglong Liu. Bifsmn: Binary neural network for keyword spotting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 4346–4352. International Joint Conferences on Artificial Intelligence Organization, 2022.

Qualcomm. Snpe: Qualcomm neural processing sdk for ai, 2022. URL https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk.

Adnan Siraj Rakin, Li Yang, Jingtao Li, Fan Yao, Chaitali Chakrabarti, Yu Cao, Jae-sun Seo, and Deliang Fan. Ra-bnn: Constructing robust & accurate binary neural network to simultaneously defend adversarial bit-flip attack and improve accuracy. *arXiv preprint arXiv:2103.13813*, 2021.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 91–99. Curran Associates, Inc., 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE CVPR*, June 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Mahendra Pratap Singh and Manoj Kumar Jain. Evolution of processor architecture in mobile phones. *International Journal of Computer Applications*, 90(4):34–39, 2014.

Vincent Vanhoucke, Andrew W. Senior, and Mark Z. Mao. Improving the speed of neural networks on cpus. 2011.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2018.

Ziwei Wang, Ziyi Wu, Jiwen Lu, and Jie Zhou. Bidet: An efficient binarized object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2049–2058, 2020.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Wikipedia. Apple m1 - wikipedia, 2022a. URL https://en.wikipedia.org/wiki/Apple_M1.

Wikipedia. Raspberry pi - wikipedia, 2022b. URL https://en.wikipedia.org/wiki/Raspberry_Pi.

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *IEEE CVPR*, June 2016.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Yixing Xu, Kai Han, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Learning frequency domain approximation for binary neural networks. *Advances in Neural Information Processing Systems*, 34:25553–25565, 2021a.

Zheng Xu, Yenchang Hsu, and Jiawei Huang. Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks. In *ICLR*, 2018.

Zihan Xu, Mingbao Lin, Jianzhuang Liu, Jie Chen, Ling Shao, Yue Gao, Yonghong Tian, and Rongrong Ji. Recu: Reviving the dead weights in binary neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5198–5208, 2021b.

Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 111–120, 2019.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *IEEE CVPR*, 2017.

Chunyu Yuan and Sos S Agaian. A comprehensive review of binary neural network. *arXiv preprint arXiv:2110.06804*, 2021.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

Chongzhi Zhang, Mingyuan Zhang, Shanghang Zhang, Daisheng Jin, Qiang Zhou, Zhongang Cai, Haiyu Zhao, Xianglong Liu, and Ziwei Liu. Delving deep into the generalization of vision transformers under distribution shifts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7277–7286, 2022.

Jianhao Zhang, Yingwei Pan, Ting Yao, He Zhao, and Tao Mei. dabnn: A super fast inference framework for binary neural networks on arm devices. In *Proceedings of the 27th ACM international conference on multimedia*, pp. 2272–2275, 2019.

Shiliang Zhang, Cong Liu, Hui Jiang, Si Wei, Lirong Dai, and Yu Hu. Feedforward sequential memory networks: A new structure to learn long-term dependency. *arXiv preprint arXiv:1512.08301*, 2015.

Shiliang Zhang, Ming Lei, Zhijie Yan, and Lirong Dai. Deep-fsmn for large vocabulary continuous speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5869–5873. IEEE, 2018a.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE CVPR*, pp. 6848–6856, 2018b.

Mengyi Zhao, Shuling Dai, Yanjun Zhu, Hao Tang, Pan Xie, Yue Li, Chunlei Liu, and Baochang Zhang. Pb-gcn: Progressive binary graph convolutional networks for skeleton-based action recognition. *Neurocomputing*, 501:640–649, 2022.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

APPENDIX FOR BIBENCH

A DETAILS OF BINARIZATION ALGORITHM

BNN (Courbariaux et al., 2016). During training process, BNN follows use the straight-through estimator (STE) to calculate gradient $\boldsymbol{g_x}$ that takes into account the saturation effect:

$$\texttt{sign}(\boldsymbol{x}) = \begin{cases} +1, & \text{if } \boldsymbol{x} \geq 0 \\ -1, & \text{otherwise} \end{cases} \qquad \boldsymbol{g_x} = \begin{cases} \boldsymbol{g_b}, & \text{if } \boldsymbol{x} \in (-1, 1) \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

And during inference, the computation process is expressed as

$$\boldsymbol{o} = \text{sign}(\boldsymbol{a}) \circledast \text{sign}(\boldsymbol{w}), \tag{11}$$

where $\circledast$ indicates a convolutional operation using XNOR and bitcount operations.

XNOR-Net (Rastegari et al., 2016). XNOR-Net obtained the channel-wise scaling factor $\boldsymbol{\alpha} = \frac{\|\boldsymbol{w}\|}{|\boldsymbol{w}|}$ for the weight and $\boldsymbol{\beta}$ for all sub-tensors in I (denoted by $\boldsymbol{K}$), and can approximate the convolution between input $\boldsymbol{a}$ and weight filter $\boldsymbol{w}$ mainly using binary operations:

$$\boldsymbol{o} = (\text{sign}(\boldsymbol{a}) \circledast \text{sign}(\boldsymbol{w})) \odot \boldsymbol{K}\boldsymbol{\alpha}, \tag{12}$$

where $\boldsymbol{w} \in \mathbb{R}^{c \times w \times h}$ and $\boldsymbol{a} \in \mathbb{R}^{c \times w_{\text{in}} \times h_{\text{in}}}$ denote the weight filter and input tensor, respectively. And the STE is also applied in the backward propagation of training process.

DoReFa-Net (Zhou et al., 2016). DoReFa-Net apply the following function for 1-bit weight and activation:

$$\boldsymbol{o} = (\text{sign}(\boldsymbol{a}) \circledast \text{sign}(\boldsymbol{w})) \odot \boldsymbol{\alpha}. \tag{13}$$

And the STE is also applied in the backward propagation with the full-precision gradient.

Bi-Real Net (Liu et al., 2018). Bi-Real Net propose a piece-wise polynomial function as the gradient approximation function:

$$\text{bireal}(\boldsymbol{a}) = \begin{cases} -1 & \text{if } \boldsymbol{a} < -1 \\ 2\boldsymbol{a} + \boldsymbol{a}^2 & \text{if } -1 \leqslant \boldsymbol{a} < 0 \\ 2\boldsymbol{a} - \boldsymbol{a}^2 & \text{if } 0 \leqslant \boldsymbol{a} < 1 \\ 1 & \text{otherwise} \end{cases}, \quad \frac{\partial\, \text{bireal}(\boldsymbol{a})}{\partial \boldsymbol{a}} = \begin{cases} 2 + 2\boldsymbol{a} & \text{if } -1 \leqslant \boldsymbol{a} < 0 \\ 2 - 2\boldsymbol{a} & \text{if } 0 \leqslant \boldsymbol{a} < 1 \\ 0 & \text{otherwise} \end{cases}. \tag{14}$$

And the forward propagation of Bi-Real Net is same as Eq. (13).

XNOR-Net++ (Bulat et al., 2019) XNOR-Net++ propose to re-formulate Eq. (12) as:

$$\boldsymbol{o} = (\text{sign}(\boldsymbol{a}) \circledast \text{sign}(\boldsymbol{w})) \odot \boldsymbol{\Gamma}, \tag{15}$$

and we adopts the $\boldsymbol{\Gamma}$ as following form in experiments (achieve best performance in original paper):

$$\boldsymbol{\Gamma} = \boldsymbol{\alpha} \otimes \boldsymbol{\beta} \otimes \boldsymbol{\gamma}, \quad \boldsymbol{\alpha} \in \mathbb{R}^{\boldsymbol{o}}, \boldsymbol{\beta} \in \mathbb{R}^{h_{\text{out}}}, \boldsymbol{\gamma} \in \mathbb{R}^{w_{\text{out}}}, \tag{16}$$

where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ are learnable during training.

ReActNet (Liu et al., 2020). ReActNet define an RSign as a binarization function with channel-wisely learnable thresholds:

$$\boldsymbol{x} = \text{rsign}(\boldsymbol{x}) = \begin{cases} +1, & \text{if } \boldsymbol{x} > \boldsymbol{\alpha} \\ -1, & \text{if } \boldsymbol{x} \leq \boldsymbol{\alpha} \end{cases}. \tag{17}$$

where $\boldsymbol{\alpha}$ is a learnable coefficient controlling the threshold. And the forward propagation is

$$\boldsymbol{o} = (\text{rsign}(\boldsymbol{a}) \circledast \text{sign}(\boldsymbol{w})) \odot \boldsymbol{\alpha}. \tag{18}$$

ReCU (Xu et al., 2021b). As described in their paper, ReCU is formulated as

$$\text{recu}(\boldsymbol{w}) = \max\left(\min\left(\boldsymbol{w}, Q_{(\tau)}\right), Q_{(1-\tau)}\right), \tag{19}$$

where $Q_{(\tau)}$ and $Q_{(1-\tau)}$ denote the τ quantile and $1-\tau$ quantile [55] of $\boldsymbol{w}$, respectively. And other implementations are also strictly followed by the original paper and official code.

FDA (Xu et al., 2021a). FDA computes the gradients of $\boldsymbol{o}$ in the backward propagation as:

$$\frac{\partial \ell}{\partial \mathbf{t}} = \frac{\partial \ell}{\partial \boldsymbol{o}} \boldsymbol{w}_2^\top \odot \left((\mathbf{t}\boldsymbol{w}_1) \geq 0 \right) \boldsymbol{w}_1^\top + \frac{\partial \ell}{\partial \boldsymbol{o}} \eta'(\mathbf{t}) + \frac{\partial \ell}{\partial \boldsymbol{o}} \odot \frac{4\omega}{\pi} \sum_{i=0}^{n} \cos((2i+1)\omega \mathbf{t}), \tag{20}$$

where $\frac{\partial \ell}{\partial \boldsymbol{o}}$ is the gradient from the upper layers, $\odot$ represents element-wise multiplication, and $\frac{\partial \ell}{\partial \mathrm{t}}$ is the partial gradient on t that backward propagate to the former layer. And $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are weights in the original models and the noise adaptation modules, FDA updates them as

$$\frac{\partial \ell}{\partial \boldsymbol{w}_1} = \mathbf{t}^\top \frac{\partial \ell}{\partial \boldsymbol{o}} \boldsymbol{w}_2^\top \odot \left((\mathbf{t}\boldsymbol{w}_1) \geq 0 \right), \qquad \frac{\partial \ell}{\partial \boldsymbol{w}_2} = \sigma\left(\mathbf{t}\boldsymbol{w}_1\right)^\top \frac{\partial \ell}{\partial \boldsymbol{o}}. \tag{21}$$

B Details of Tasks

CIFAR10 (Krizhevsky et al., 2014) The CIFAR-10 dataset (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. This dataset is widely used for image classification tasks. There are 60,000 color images in the CIFAR-10 dataset that are in 10 different classes, each of which measures 32x32 pixels. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6000 images per class with 5000 training and 1000 testing images per class. The evaluation metric of CIFAR-10 dataset is accuracy, defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{22}$$

where *TP* (True Positive) means cases correctly identified as positive, *TN* (True Negative) means cases correctly identified as negative, *FP* (False Positive) means cases incorrectly identified as positive and *FN* (False Negative) means cases incorrectly identified as negative. To estimate the accuracy, we should calculate the proportion of *TP* and *TN* in all evaluated cases.

ImageNet (Krizhevsky et al., 2012) ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. A crowd-sourced image labeling service called Amazon Mechanical Turk was utilized to collect the images from the web and label them by human labelers. As part of the Pascal Visual Object Challenge, ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) was established in 2010. In total, there are approximately 1.2 million training images, 50,000 validation images, and 150,000 testing images used in ILSVRC. ILSVRC uses a subset of ImageNet, with approximately 1000 images in each of 1000 categories.

ImageNet also uses accuracy to evaluate the predict results, which is defined above.

VOC07 (Hoiem et al., 2009) The PASCAL Visual Object Classes 2007 (VOC07) dataset contains 20 object categories including vehicles, household, animals, and other: aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse, sheep, and person. As a benchmark for object detection, semantic segmentation, and object classification, this dataset contains pixel-level segmentation annotations, bounding box annotations, and object class annotations. The VOC07 dataset uses mean average precision(mAP) to evaluate results, which is defined as:

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{23}$$

where AP_k denotes the average precision of the k-th category, which calculates the area under the precision-recall curve:

$$AP_k = \int_0^1 p_k(r) dr. \tag{24}$$

Specially, for VOC07, we apply 11-point interpolated AP which devides the recall value to $\{0.0, 0.1, \ldots, 1.0\}$ and then computes the average of maximum precision value for these 11 recall

values as:

$$AP = \frac{1}{11} \sum_{r \in \{0.0,\ldots,1.0\}} AP_r \tag{25}$$

$$= \frac{1}{11} \sum_{r \in \{0.0,\ldots,1.0\}} p_{\text{interp}} r. \tag{26}$$

The maximum precision value equals to the right of its recall level:

$$p_{\text{interp}} r = \max_{\tilde{r} \geq r} p(\tilde{r}). \tag{27}$$

COCO17 (Lin et al., 2014) The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images. According to community feedback, in 2017 the training/validation split was changed from 83K/41K to 118K/5K. In the 2017 release, the images and annotations are the same. The 2017 test set is a subset of 41K images from the 2015 test set. Additionally, 123K images are included in the unannotated dataset. The COCO17 dataset also uses mean average precision (mAP) as defined above PASCAL VOC07 uses, which is defined as above.

ModelNet40 (Wu et al., 2015) The ModelNet40 dataset contains synthetic point clouds of synthetic objects. As the most widely used benchmark for point cloud analysis, ModelNet40 is popular due to the diversity of categories, clean shapes, and well-constructed dataset. In the original ModelNet40, 12,311 CAD-generated meshes are divided into 40 categories, of which 9,843 are used for training and 2,468 are used for testing. Using a uniform sampling method from mesh surfaces, the point cloud data points are uniformly sampled and then scaled into a unit sphere by moving to the origin and moving to the origin. The ModelNet40 dataset also uses accuracy as metric which is defined above in CIFAR10.

ShapeNet (Chang et al., 2015) ShapeNet is a large scale repository for 3D CAD models developed by researchers from Stanford University, Princeton University and the Toyota Technological Institute at Chicago, USA.

Using WordNet hypernym-hyponym relationships, the repository contains over 300M models, with 220,000 of them classified into 3,135 classes. There are 31,693 meshes in the ShapeNet Parts subset, divided into 16 classes of objects (i.e. tables, chairs, planes, etc.). Each shape contains 2-5 parts (with 50 part classes in total).

GLUE (Wang et al., 2018) General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding tasks, including single-sentence tasks CoLA and SST-2, similarity and paraphrasing tasks MRPC, STS-B and QQP, and natural language inference tasks MNLI, QNLI, RTE and WNLI. Among them, SST-2, MRPC, QQP, MNLI, QNLI, RTE, WNLI use accuracy as metric which is defined in CIFAR10, CoLA is measured by Matthews Correlation Coefficient (MCC) which is better in binary classification, especially the number of positive and negative samples are extremely unbalanced:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{28}$$

And STS-B is measured by Pearson/Spearman Correlation Coefficient:

$$r_{Pearson} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\frac{X_i - \bar{X}}{s_X} \right) \left(\frac{Y_i - \bar{Y}}{s_Y} \right), r_{Spearman} = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}, \tag{29}$$

where n is the number of observations, s_X and s_Y indicate the sum of squares of X and Y respectively, and d_i is the difference between the ranks of corresponding variables.

SpeechCom. (Warden, 2018) As part of its training and evaluation process, SpeechCom provides a collection of audio recordings containing spoken words. Its primary goal is to provide a way to build and test small models that detect when a single word is spoken from a set of ten target words, with as few false positives as possible from background noise or unrelated speech, while providing as few false positives as possible. The accuracy metric for SpeechCom is also the same as CIFAR10.

CIFAR10-C (Hendrycks & Dietterich, 2018) CIFAR10-C is a dataset generated by adding 15 common corruptions and 4 extra corruptions to the test images in the Cifar10 dataset. It benchmarks the frailty of classifiers under corruptions, including noise, blur, weather and digital influence. And each type of corruption has five level of severity, resulting in 75 distinct corruptions. We report the accuracy of the classifiers under each level of severity and each corruption. Meanwhile, we use the mean and relative corruption error as metrics. Denote the error rate of *Network* under *Settings* as $E^{Network}_{Settings}$. The classifier's aggregate performance across the five severities of the corruption types. The Corruption Errors of a certain type of *Corruption* is computed with the formula:

$$CE^{Network}_{Corruption} = \sum_{s=1}^{5} E^{Network}_{s,Corruption} / \sum_{s=1}^{5} E^{AlexNet}_{s,Corruption}. \tag{30}$$

To make Corruption Errors comparable across corruption types, the difficulty is usually adjusted by dividing by AlexNet's errors.

C Details of Architectures

ResNet (He et al., 2016) Residual Networks, or ResNets, learn residual functions concerning the layer inputs instead of learning unreferenced functions. Instead of hoping every few stacked layers directly fit a desired underlying mapping, residual nets let these layers fit a residual mapping. There is empirical evidence that these types of networks are easier to optimize and can gain accuracy from considerably increased depth.

VGG (Simonyan & Zisserman, 2015) VGG is a classical convolutional neural network architecture. It was based on an analysis of how to increase the depth of such networks. The network utilizes small 3 x 3 filters. Otherwise, the network is characterized by its simplicity: the only other components being pooling layers and a fully connected layer.

MobileNetV2 is a convolutional neural network architecture designed to perform well on mobile devices. This model is based on an inverted residual structure with residual connections between the bottleneck layers. As a source of nonlinearity, the intermediate expansion layer employs lightweight depthwise convolutions to filter features. In MobileNetV2, the architecture begins with an initial layer of 32 convolution filters, followed by 19 residual bottleneck layers.

Faster-RCNN (Ren et al., 2015) Faster R-CNN is an object detection model that improves Fast R-CNN by utilizing a region proposal network (RPN) with the CNN model. The RPN shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals. A fully convolutional network is used to predict the bounds and objectness scores of objects at each position simultaneously. RPNs are trained end-to-end to produce region proposals of high quality, which Fast R-CNN uses for detection. By sharing their convolutional features, RPN and Fast R-CNN are combined into a single network: the RPN component instructs the unified network on where to search. There are two modules that make up Faster R-CNN. The first module is a deep, fully convolutional network that proposes regions, and the second is the Fast R-CNN detector that uses the proposed regions.

SSD (Liu et al., 2016) SSD is a single-stage object detection method that discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. During prediction, each default box is adjusted to better match the shape of the object based on its scores for each object category. In addition, the network automatically handles objects of different sizes by combining predictions from multiple feature maps with different resolutions.

BERT (Kenton & Toutanova, 2019) BERT, or Bidirectional Encoder Representations from Transformers, improves upon standard Transformers by removing the unidirectionality constraint using a masked language model (MLM) pre-training objective. By masking some tokens from the input, the masked language model attempts to estimate the original vocabulary id of the masked word based solely on its context. A MLM objective differs from a left-to-right language model in that it enables the representation to integrate the left and right contexts, which facilitates the pre-training of a deep bidirectional Transformer. Additionally, BERT uses a next sentence prediction task that pre-trains text-pair representations along with the masked language model.

PointNet (Qi et al., 2017) PointNet is a unified architecture for applications ranging from object classification and part segmentation to scene semantic parsing. The architecture directly receives

point clouds as input and outputs either class labels for the entire input or point segment/part labels. **PointNet-Vanilla** is a variant of PointNet, which drops off the T-Net module.

FSMN (Zhang et al., 2015) Feedforward sequential memory networks or FSMN is a novel neural network structure to model long-term dependency in time series without using recurrent feedback. It is a standard fully connected feedforward neural network containing some learnable memory blocks. As a short-term memory mechanism, the memory blocks encode long context information using a tapped-delay line structure.

Deep-FSMN (Zhang et al., 2018a) The Deep-FSMN architecture is an improved feedforward sequential memory network (FSMN) with skip connections between memory blocks in adjacent layers. By utilizing skip connections, information can be transferred across layers and thus the gradient vanishing problem can be avoided when building very deep structures.

D Details of Hardwares

Hisilicon Kirin (Hisilicon, 2022). Kirin is a series of ARM-based systems-on-a-chip (SoCs) produced by HiSilicon. Their products include Kirin 970, Kirin 980, Kirin 985, etc.

MediaTek Dimensity (MediaTek, 2022). Dimensity is a series of ARM-based systems-on-a-chip (SoCs) produced by MediaTek. Their products include Dimensity 820, Dimensity 8100, Dimensity 9000, etc.

Qualcomm Snapdragon (Singh & Jain, 2014). Snapdragon is a family of mobile systems-on-a-chip (SoC) processor architecture provided by Qualcomm. The original Snapdragon chip, the Scorpio, was similar to the ARM Cortex-A8 core based upon the ARMv7 instruction set, but it was enhanced by the use of SIMD operations, which provided higher performance. Qualcomm Snapdragon processors are based on the Krait architecture. They are equipped with an integrated LTE modem, which provides seamless connectivity across both 2G and 3G LTE networks.

Raspberrypi (Wikipedia, 2022b). Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. Raspberry Pi was originally designed to promote the teaching of basic computer science in schools and in developing countries. As a result of its low cost, modularity, and open design, it is used in many applications, including weather monitoring, and is sold outside the intended market. It is typically used by computer hobbyists and electronic enthusiasts due to the adoption of HDMI and USB standards.

Apple M1 (Wikipedia, 2022a). Apple M1 is a series of ARM-based systems-on-a-chip (SoCs) designed by Apple Inc. As a central processing unit (CPU) and graphics processing unit (GPU) for Macintosh desktops and notebooks, as well as iPad Pro and iPad Air tablets. In November 2020, Apple introduced the M1 chip, followed by the professional-oriented M1 Pro and M1 Max chips in 2021. Apple launched the M1 Ultra in 2022, which combines two M1 Max chips in a single package. The M1 Max is a higher-performance version of the M1 Pro, with larger GPU cores and memory bandwidth.

E Full Results

Table 5-6 shows accuracy of different binarization algorithms on 2D and 3D vision tasks, including CIFAR10, ImageNet, PASCAL VOC07, COCO14 for 2D vision tasks and ModelNet40 for 3D vision task. And for each task, we cover several representative model architectures, and binarize them with aforementioned binarization algorithms.

We also evaluate binarization algorithms on language and speech tasks, for which we test TinyBERT (4 layers and 6 layers) on GLUE Benchmark and FSMN and D-FSMN on Speech Command dataset. Results are listed as Table 7.

To demonstrate the robustness corruption of binarized algorithms, we show the results on CIFAR10-C benchmark which is used to benchmark the robustness to common perturbations in Table 8 and Table 9. It includes 15 kinds of noise, blur, weather, digital corruptions, and each type of corruption has five levels of severity.

The sensitivity of hyper parameters while training is shown in Table 10-11. For each binarization algorithm, we use SGD or Adam optimizer, $1\times$ or $0.1\times$ of the original learning rate, cosine or step learning scheduler, and we train 200 epochs. Each case is tested five times to show the training stability. We also calculate the mean and standard deviation (std) of accuracy. We bold the best accuracy and the lowest std for each binarization algorithm.

We conduct comprehensive deployment and inference on various kinds of hardwares, include Kirin series (970, 980, 985, 990 and 9000E), Dimensity series (820 and 9000), Snapdragon series (855+, 870 and 888), Raspberrypi (3B+ and 4B) and Apple M1 series (M1 and M1 Max). Limited to the support of frameworks, we can only test BNN and ReAct with Larq compute engine and only BNN with daBNN. We convert models to enable the actual inference on real hardware, include ResNet18/34 and VGG-Small on Larq, and only ResNet18/34 on daBNN. And we test 1, 2, 4, 8 threads for each hardware and additionally test 16 threads for Apple Silicons on Larq. And daBNN only support single thread inference. Results are showcased in Table 12-15.

Table 5: Accuracy on 2D and 3D Vision Tasks.

Task	Arch.	FP32	Binarization Algorithm							
			BNN	XNOR	DoReFa	Bi-Real	XNOR++	ReActNet	ReCU	FDA
CIFAR10	ResNet20	91.99	85.31	85.53	85.18	85.56	85.41	86.18	86.42	86.38
	ResNet18	94.82	89.69	91.4	91.55	91.20	90.04	91.55	92.79	90.42
	ResNet34	95.34	90.82	89.58	90.95	92.50	90.59	92.69	93.64	89.59
	VGG-Small	93.80	89.66	89.65	89.66	90.25	89.34	90.27	90.84	89.48
ImageNet	ResNet18	69.90	52.99	53.99	53.55	54.79	52.43	54.97	54.51	54.63
VOC07	Faster-RCNN	76.06	58.54	56.75	58.07	60.90	56.60	61.90	62.10	60.10
	SSD300	77.34	9.09	33.72	30.70	31.90	9.41	38.41	9.80	43.68
COCO14	Faster-RCNN	27.20	21.20	20.50	21.30	22.20	21.60	22.80	23.30	22.40
ModelNet40	$\text{PointNet}_{\text{vanilla}}$	86.80	85.13	83.47	85.21	85.37	85.66	85.13	85.21	85.49
	PointNet	88.20	9.08	80.75	78.77	77.71	63.25	76.50	81.12	79.62

Table 6: Accuracy on ShapeNet dataset.

Task	Arch.	Category	FP32	Binarization Algorithm							
				BNN	XNOR	DoReFa	Bi-Real	XNOR++	ReActNet	ReCU	FDA
ShapeNet	PointNet	Airplane	83.7	37.5	74.14	67.2	67.61	30.36	66.12	31.61	65.34
		Bag	79.6	44.2	49	55.34	47.11	37.44	50.28	38.58	48.62
		Cap	92.3	44.3	73.32	51.21	61.41	40.37	56.73	40.13	56
		Car	76.8	24.3	55.27	52.24	49.39	24.07	49.11	23.92	58.5
		Chair	90.9	61.6	85.62	83.96	83.6	41.89	83.83	41.5	83.27
		Earphone	70.2	38.5	30.97	34.94	35.24	26.3	36.72	23.01	34.46
		Guitar	91.1	32.9	69.17	67.9	65.99	23.45	64.18	28.38	78.69
		Knife	85.7	43	78.16	76.16	75.53	37.62	75.01	38.81	77.07
		Lamp	82	51.2	69	68.75	60	49.45	66.13	48.41	67.45
		Laptop	95.5	49.4	93.29	92.93	92.79	41.89	92.93	42.28	93.66
		Motorbike	64.4	16.3	19.04	18.88	18.69	13.18	18.59	11.26	20.38
		Mug	93.6	49.1	64.32	53.56	52.01	47.58	52.51	46.83	53.48
		Pistol	80.8	25.5	62.29	59.15	51.43	26.96	53.79	27.81	62.61
		Rocket	54.4	26.9	30.95	27.92	26.61	22.38	26.01	19.32	23.08
		Skateboard	70.7	41.2	45.7	50.15	43.78	28.63	43.74	26.71	45.81
		Table	81.4	51.3	73.68	72.69	69.72	45.74	69.56	45.21	73.45
		Overall	80.81875	39.82	60.87	58.31	56.31	33.58	56.58	33.36	58.68

Table 7: Accuracy on Language and Speech Tasks.

Task		Arch.	FP32	Binarization Algorithm							
				BNN	XNOR	DoReFa	Bi-Real	XNOR++	ReActNet	ReCU	FDA
GLUE	MNLI-m	BERT-Tiny$_{4L}$	82.81	36.90	41.20	52.31	55.09	37.27	55.52	38.55	59.41
		BERT-Tiny$_{6L}$	84.76	37.01	51.17	63.09	66.81	37.98	66.47	37.95	68.46
		BERT-Base	84.88	35.45	41.40	60.67	62.47	35.45	60.22	35.45	63.49
	MNLI-mm	BERT-Tiny$_{4L}$	83.08	36.54	41.55	53.01	55.57	36.07	55.89	37.62	59.76
		BERT-Tiny$_{6L}$	84.42	36.47	50.92	63.87	66.82	38.11	67.64	36.91	68.98
		BERT-Base	85.45	35.22	41.18	60.96	63.17	35.22	61.19	35.22	63.72
	QQP	BERT-Tiny$_{4L}$	90.47	66.19	73.69	75.79	77.38	64.97	76.92	67.32	78.92
		BERT-Tiny$_{6L}$	85.98	63.18	78.90	80.93	82.42	63.19	82.95	63.3	83.19
		BERT-Base	91.51	63.18	71.93	77.07	80.01	63.18	81.16	63.18	83.26
	QNLI	BERT-Tiny$_{4L}$	87.46	51.71	60.59	61.15	61.92	52.79	62.67	53.99	62.29
		BERT-Tiny$_{6L}$	90.79	52.22	62.75	66.88	69.72	51.84	70.27	51.32	72.72
		BERT-Base	92.14	51.8	60.29	70.78	70.14	54.07	69.44	51.87	72.43
	SST-2	BERT-Tiny$_{4L}$	92.43	52.98	79.93	82.45	84.06	54.01	84.17	54.24	86.12
		BERT-Tiny$_{6L}$	90.25	58.14	84.74	86.23	87.73	69.38	87.95	52.40	87.72
		BERT-Base	93.23	52.29	78.78	86.01	86.35	53.32	84.4	52.40	87.93
	CoLA	BERT-Tiny$_{4L}$	49.61	6.55	7.22	12.69	16.86	0	14.71	6.25	17.80
		BERT-Tiny$_{6L}$	54.17	2.57	12.57	15.97	17.94	0	15.24	2.24	22.21
		BERT-Base	59.71	4.63	0	4.74	15.95	0	4.63	0.40	4.63
	STS-B	BERT-Tiny$_{4L}$	86.35	4.31	18.05	18.74	22.65	7.45	22.73	8.20	27.56
		BERT-Tiny$_{6L}$	89.79	1.04	14.72	22.31	24.59	5.70	23.40	8.22	37.15
		BERT-Base	90.06	6.94	12.19	18.26	20.76	4.99	8.73	6.59	10.14
	MRPC	BERT-Tiny$_{4L}$	85.50	68.30	71.74	71.99	71.74	68.30	71.74	71.25	71.49
		BERT-Tiny$_{6L}$	87.71	68.30	70.76	71.74	71.49	68.30	71.74	69.04	71.74
		BERT-Base	86.24	68.30	68.3	70.02	70.27	68.30	71.25	68.30	69.04
	RTE	BERT-Tiny$_{4L}$	65.34	56.31	53.43	56.31	55.59	54.15	57.76	61.01	59.20
		BERT-Tiny$_{6L}$	68.95	56.31	54.51	54.51	58.12	49.09	53.43	58.84	54.87
		BERT-Base	72.20	53.43	57.04	55.23	54.51	54.87	54.51	55.23	55.23
Speech Commands		FSMN	94.89	56.45	56.45	68.65	73.60	75.04	73.80	56.45	56.45
		D-FSMN	97.51	88.32	92.03	78.92	85.11	56.77	83.80	92.11	93.91

Table 8: Results for Robustness Corruption on CIFAR10-C Dataset with Different Binarization Algorithms (1/2).

Noise	FP32	Binarization Algorithm							
		BNN	XNOR	DoReFa	Bi-Real	XNOR++	ReActNet	ReCU	FDA
Origin	94.82	89.69	91.40	91.55	91.20	90.04	91.55	92.79	90.42
gaussian_noise-1	78.23	74.22	76.00	74.97	74.95	75.07	75.15	78.25	77.36
gaussian_noise-2	56.72	56.73	62.44	55.94	58.33	57.52	55.97	61.32	60.48
gaussian_noise-3	36.93	42.69	47.58	39.56	43.47	40.79	37.99	43.32	44.26
gaussian_noise-4	31.03	38.35	41.43	33.24	36.65	34.68	31.47	35.91	37.30
gaussian_noise-5	25.54	34.05	36.22	28.66	31.78	30.13	25.49	30.19	32.09
ipulse_nosie-1	82.54	84.57	86.94	84.68	87.30	85.72	86.89	88.73	85.8
ipulse_nosie-2	70.12	77.13	80.74	77.35	81.14	79.62	80.25	82.80	80.3
ipulse_nosie-3	59.88	70.58	75.01	69.20	74.59	71.82	72.16	76.05	72.6
ipulse_nosie-4	40.59	54.42	59.39	49.48	56.66	52.61	49.79	58.44	56.45
ipulse_nosie-5	26.03	39.86	41.54	32.72	37.28	35.12	28.42	38.26	39.98
shot_noise-1	85.75	81.51	81.31	81.84	81.58	80.66	81.88	83.98	82.42
shot_noise-2	76.61	72.04	74.02	72.21	72.81	73.03	72.32	76.70	75.1
shot_noise-3	52.21	53.90	57.08	50.66	54.59	53.76	51.31	57.22	56.56
shot_noise-4	44.13	47.58	51.29	43.59	48.36	46.64	44.21	48.78	48.91
shot_noise-5	32.73	39.93	40.79	33.80	38.50	36.47	31.79	36.46	37.8
speckle_noise-1	86.30	81.29	81.94	80.93	80.77	81.14	82.17	84.17	82.62
speckle_noise-2	71.94	68.07	70.14	67.5	69.22	69.35	68.26	72.94	71.70
speckle_noise-3	64.47	62.12	64.13	60.24	63.44	62.50	61.14	66.89	64.27
speckle_noise-4	49.81	51.93	53.77	47.93	52.75	50.59	48.39	54.13	52.40
speckle_noise-5	38.70	44.25	43.60	38.65	43.16	42.09	37.78	42.13	42.57
gaussian_blur-1	94.17	89.03	90.5	89.33	90.56	89.00	91.05	92.16	89.33
gaussian_blur-2	87.04	78.3	81.98	78.81	80.42	77.75	81.20	84.80	78.93
gaussian_blur-3	75.15	67.74	68.27	67.67	68.16	64.54	67.42	73.62	66.29
gaussian_blur-4	59.5	55.17	53.63	55.74	54.08	52.44	52.72	60.32	53.37
gaussian_blur-5	36.03	37.31	33.96	37.50	37.54	36.77	34.08	39.22	34.93
defocus_blur-1	94.2	88.73	91.06	89.1	90.32	88.91	90.92	91.98	89.58
defocus_blur-2	92.75	85.97	88.99	86.59	88.31	85.58	87.91	90.47	87.01
defocus_blur-3	87.38	79.02	82.43	78.88	80.71	77.58	80.88	84.85	79.52
defocus_blur-4	76.99	69.13	71.02	68.29	68.33	65.96	68.42	74.40	68.22
defocus_blur-5	52.09	48.85	51.99	48.82	49.17	48.45	46.92	55.70	48.27
glass_blur-1	54.93	56.57	51.72	57.94	56.78	57.29	56.27	58.82	58.92
glass_blur-2	56.37	57.93	53.46	60.42	59.21	59.32	58.03	60.25	60.56
glass_blur-3	59.21	61.43	56.98	64.11	61.72	62.41	60.39	62.84	63.32
glass_blur-4	45.65	46.50	42.72	48.48	47.19	47.83	46.88	49.09	49.23
glass_blur-5	49.19	49.52	46.40	52.06	49.83	50.02	49.08	51.14	51.82
otion_blur-1	89.40	81.57	83.27	82.00	83.11	81.48	84.19	86.21	82.83
otion_blur-2	81.95	71.52	74.71	73.38	72.48	70.99	74.35	77.75	74.09
otion_blur-3	72.48	61.87	66.21	63.86	63.39	61.57	63.85	68.31	64.36
otion_blur-4	72.79	62.40	66.18	63.94	62.84	62.03	64.54	67.88	64.13
otion_blur-5	63.91	54.14	57.98	56.07	55.90	54.35	55.60	59.71	56.65
zoo_blur-1	87.36	78.25	81.31	78.56	79.45	77.20	80.22	83.69	78.55
zoo_blur-2	83.89	74.84	77.73	75.39	75.88	72.83	75.74	80.46	74.72
zoo_blur-3	77.73	69.00	70.98	68.81	69.03	66.56	68.34	74.33	68.21
zoo_blur-4	71.39	64.12	65.21	63.79	62.81	61.01	62.47	68.67	62.58
zoo_blur-5	60.60	55.15	55.83	55.4	54.38	52.66	52.24	59.51	53.94
brighness-1	94.31	89.29	90.84	89.53	90.8	89.30	90.97	92.06	89.74
brighness-2	94.03	88.25	90.42	88.71	89.66	88.50	90.64	91.64	88.77
brighness-3	93.53	87.40	89.38	87.38	89.17	87.31	89.63	90.72	87.84
brighness-4	92.74	85.45	88.27	86.12	87.78	85.44	88.16	89.58	86.39
brighness-5	90.36	80.95	85.22	81.65	83.79	80.99	84.45	86.53	82.04

Table 9: Results for Robustness Corruption on CIFAR10-C Dataset with Different Binarization Algorithms (2/2).

Noise	FP32	Binarization Algorithm							
		BNN	XNOR	DoReFa	Bi-Real	XNOR++	ReActNet	ReCU	FDA
fog-1	94.04	88.17	90.89	88.84	89.91	88.51	90.84	92.08	89.43
fog-2	93.03	84.58	88.85	85.48	87.26	84.77	88	89.87	86.76
fog-3	90.69	78.07	85.2	80.07	83.32	78.94	83.77	86.82	82.78
fog-4	86.72	69.56	78.92	72.27	75.89	71.01	77.96	81.56	75.62
fog-5	68.6	49.04	53.9	52.33	52.88	49.68	57.67	62.29	55.18
frost-1	89.97	83.66	84.7	84.07	85.76	83.64	85.51	87.75	84.85
frost-2	84.42	77.88	78.97	77.47	78.96	77.26	79.14	81.4	79.16
frost-3	74.85	67.67	69.3	67.14	68.76	66.03	69.58	72.54	70.14
frost-4	73.32	65.93	67.14	65.97	68.52	65.37	67.93	71.44	69.41
frost-5	62.13	55.02	56.67	55.62	56.77	54.26	57.69	61.11	59.88
snow-1	89.26	84.58	85.44	84.59	86.43	85.07	85.95	87.82	85.67
snow-2	78.96	72.01	73.37	73.14	73.42	72.43	73.65	78.05	73.84
snow-3	82.85	75.6	76.84	75.74	76.95	75.74	77.76	79.98	75.95
snow-4	80.29	70.84	72.56	71.93	72.39	71.25	72.97	76.12	72.16
snow-5	74.94	63.85	66.94	65.9	65.71	64.24	66.29	70.33	66.53
contrast-1	93.82	87.23	89.96	87.93	89.68	87.88	90.16	91.53	88.84
contrast-2	90.53	76.02	84	77.27	82.1	76.3	82.8	86.02	81.37
contrast-3	85.84	63.97	77.1	65.77	72.77	64.18	74.62	79.30	71.89
contrast-4	75.08	44.07	62.37	47.35	55.57	44.94	59.87	65.72	55.14
contrast-5	29.36	20	25.67	20.18	22.18	21.04	25.28	25.66	24.04
elastic_transfor-1	89.97	82.97	84.5	83.38	84.74	82.48	84.42	86.54	83.25
elastic_transfor-2	89.43	82.12	84.79	82.44	84.07	81.97	84.61	86.20	83.15
elastic_transfor-3	85.52	77.56	80.71	77.92	79.11	77	79.53	82.27	78.35
elastic_transfor-4	79.48	73.75	74.83	73.92	73.41	72.77	73.98	77.53	74.17
elastic_transfor-5	75.02	70.97	70.22	71.36	71.03	71.63	71.65	75.31	71.49
jpeg_copression-1	87.36	83.28	83.93	84.07	83.83	83.77	84.3	85.65	83.63
jpeg_copression-2	81.68	80.09	79.66	79.77	80.03	80.29	80.59	81.66	79.8
jpeg_copression-3	79.98	78.55	78.21	78.32	78.27	78.99	78.51	79.94	78.44
jpeg_copression-4	77.17	77.12	75.78	77.44	77.04	77.46	77.08	77.67	76.71
jpeg_copression-5	73.85	74.51	73.04	74.65	74.13	75.26	74.16	74.85	74.19
pixelate-1	92.57	86.97	88.19	87.39	88.73	87.47	88.17	89.42	87.26
pixelate-2	88.23	81.91	80.95	82.37	82.82	81.93	81.99	83.80	80.95
pixelate-3	84	78.4	75.25	78.63	78.03	77.15	77.28	78.89	75.30
pixelate-4	68.51	64.11	58.11	62.49	60.89	61.43	60.06	61.71	59.95
pixelate-5	50.57	50.68	44.77	48.18	45.44	46.74	43.27	47.29	45.62
saturate-1	92.41	84.98	88.38	85.38	87.26	85.39	88.23	89.34	86.82
saturate-2	90.12	80.74	85.26	81.57	82.68	80.74	84.22	86.06	83.09
saturate-3	93.83	87.89	90.45	88.12	89.53	88.03	90.15	90.97	88.73
saturate-4	91.61	82.5	86.88	82.6	84.66	82.44	86.04	87.56	83.70
saturate-5	87.48	76.03	82.76	75.53	78.3	75.85	80.62	82.64	77.00
spatter-1	91.17	87.5	89.75	87.83	89.34	87.9	89.42	91.00	88.98
spatter-2	85.2	83.85	85.98	83.52	85.64	84.72	85.88	87.59	85.00
spatter-3	80.63	77.94	80.33	77.95	80.19	79.41	80.07	82.65	79.50
spatter-4	94.68	84.57	86.71	84.77	86.51	85.14	86.72	88.22	85.32
spatter-5	74.07	78.85	80.77	78.71	80.94	79.71	80.51	83.14	79.48
Overall	74.11	69.43	71.51	69.36	70.76	69.09	70.31	73.56	70.70

Table 10: Sensitivity to Hyper Parameters in Training (1/2).

Algorithm	Epoch	Optimizer	Learning Rate	Scheduler	Acc.	Acc._1	Acc._2	Acc._3	Acc._4	mean	std
FP32	200	SGD	0.1	cosine	94.58	94.6	95.05	94.64	94.84	94.74	0.20
	200	SGD	0.1	step	92.63	92.42	92.15	92.62	92.38	92.44	0.20
	200	SGD	0.01	cosine	92.23	91.76	91.76	91.99	92.17	91.98	0.22
	200	SGD	0.01	step	83.94	83.50	82.80	84.13	83.89	83.65	0.53
	200	Adam	0.001	cosine	93.51	92.94	93.12	93.35	92.86	93.16	0.27
	200	Adam	0.001	step	93.37	93.15	93.32	93.41	93.35	93.32	0.10
	200	Adam	0.0001	cosine	89.97	89.92	89.96	89.9	89.92	89.93	0.03
	200	Adam	0.0001	step	90.57	89.91	90.43	90.25	90.31	90.29	0.25
BNN	200	SGD	0.1	cosine	87.62	87.53	87.99	88.86	87.84	87.97	0.53
	200	SGD	0.1	step	70.87	73.86	71.83	73.1	72.87	72.51	1.17
	200	SGD	0.01	cosine	73.52	72.62	72.82	71.14	72.59	72.54	0.87
	200	SGD	0.01	step	52.85	51.77	52.00	52.34	53.14	52.42	0.57
	200	Adam	0.001	cosine	88.76	88.99	88.67	88.84	88.81	88.81	0.12
	200	Adam	0.001	step	88.85	89.34	88.77	89.02	89.00	89.00	0.22
	200	Adam	0.0001	cosine	83.46	83.09	83.20	83.70	83.20	83.33	0.25
	200	Adam	0.0001	step	84.08	84.11	84.20	84.31	83.56	84.05	0.29
XNOR	200	SGD	0.1	cosine	91.83	91.99	91.87	92.01	91.56	91.85	0.18
	200	SGD	0.1	step	90.02	90.01	90.12	89.82	89.7	89.93	0.17
	200	SGD	0.01	cosine	90.09	89.68	90.01	89.97	90.00	89.95	0.16
	200	SGD	0.01	step	86.86	86.66	87.21	86.98	86.61	86.86	0.24
	200	Adam	0.001	cosine	89.39	89.81	89.73	89.91	89.75	89.72	0.20
	200	Adam	0.001	step	89.92	89.79	89.73	90.01	89.61	89.81	0.16
	200	Adam	0.0001	cosine	86.18	86.29	87.03	86.36	86.62	86.50	0.34
	200	Adam	0.0001	step	86.32	87.04	86.68	86.99	87.18	86.84	0.34
DoReFa	200	SGD	0.1	cosine	85.64	85.67	85.89	86.00	85.79	85.80	0.15
	200	SGD	0.1	step	86.95	86.98	86.69	86.62	86.65	86.78	0.17
	200	SGD	0.01	cosine	86.56	86.59	86.52	86.69	86.88	86.65	0.14
	200	SGD	0.01	step	78.76	79.97	80.73	79.94	80.47	79.97	0.76
	200	Adam	0.001	cosine	88.85	89.06	88.92	88.87	88.75	88.89	0.11
	200	Adam	0.001	step	89.08	89.16	88.93	89.23	88.84	89.05	0.16
	200	Adam	0.0001	cosine	83.56	83.17	83.65	83.60	83.66	83.53	0.20
	200	Adam	0.0001	step	83.70	83.74	84.27	84.19	84.01	83.98	0.26
Bi-Real	200	SGD	0.1	cosine	87.55	87.81	88.06	87.30	87.88	87.72	0.30
	200	SGD	0.1	step	87.95	88.35	88.13	87.73	88.25	88.08	0.25
	200	SGD	0.01	cosine	87.76	87.93	87.73	87.72	87.64	87.76	0.11
	200	SGD	0.01	step	83.75	82.91	82.82	82.91	83.39	83.16	0.40
	200	Adam	0.001	cosine	88.78	89.15	89.06	89.00	89.2	89.04	0.16
	200	Adam	0.001	step	88.89	88.98	88.78	89.11	89.05	88.96	0.13
	200	Adam	0.0001	cosine	83.96	84.17	84.37	83.54	84.07	84.02	0.31
	200	Adam	0.0001	step	84.63	84.48	84.32	84.75	84.29	84.49	0.20

Table 11: Sensitivity to Hyper Parameters in Training (2/2).

Algorithm	Epoch	Optimizer	Learning Rate	Scheduler	Acc.	Acc.$_1$	Acc.$_2$	Acc.$_3$	Acc.$_4$	mean	std
XNOR++	200	SGD	0.1	cosine	87.82	88.42	88.12	88.55	88.19	88.22	0.28
	200	SGD	0.1	step	73.55	73.11	75.06	74.05	73.78	73.91	0.73
	200	SGD	0.01	cosine	74.03	75.06	73.64	74.53	74.71	74.39	0.56
	200	SGD	0.01	step	53.55	54.16	54.01	52.91	54.36	53.80	0.58
	200	Adam	0.001	cosine	88.77	88.65	89.10	88.61	88.81	88.79	0.19
	200	Adam	0.001	step	89.18	89.05	89.27	88.93	89.00	89.09	0.14
	200	Adam	0.0001	cosine	83.86	83.49	83.56	83.16	83.62	83.54	0.25
	200	Adam	0.0001	step	83.46	83.77	84.40	84.06	83.82	83.90	0.35
ReActNet	200	SGD	0.1	cosine	88.60	88.53	88.38	88.48	88.89	88.58	0.19
	200	SGD	0.1	step	88.42	88.01	88.10	88.02	88.43	88.20	0.21
	200	SGD	0.01	cosine	87.75	87.86	88.00	87.80	88.02	87.89	0.12
	200	SGD	0.01	step	83.29	82.89	83.65	83.76	83.27	83.37	0.35
	200	Adam	0.001	cosine	89.47	89.29	89.01	89.05	89.14	89.19	0.19
	200	Adam	0.001	step	89.27	89.74	89.48	89.40	89.39	89.46	0.18
	200	Adam	0.0001	cosine	84.65	84.93	84.48	84.65	84.67	84.68	0.16
	200	Adam	0.0001	step	84.69	84.55	84.93	84.94	85.38	84.90	0.32
ReCU	200	SGD	0.1	cosine	91.72	91.94	91.68	91.69	91.81	91.77	0.11
	200	SGD	0.1	step	87.73	88.14	87.81	88.02	87.91	87.92	0.16
	200	SGD	0.01	cosine	87.32	87.28	87.53	87.48	87.32	87.39	0.11
	200	SGD	0.01	step	71.86	71.72	71.78	72.26	71.59	71.84	0.25
	200	Adam	0.001	cosine	88.24	89.98	88.26	88.48	88.13	88.62	0.77
	200	Adam	0.001	step	88.36	88.48	88.55	88.42	88.63	88.49	0.11
	200	Adam	0.0001	cosine	80.07	81.10	80.62	81.09	79.95	80.57	0.55
	200	Adam	0.0001	step	81.26	81.42	81.08	81.58	81.69	81.41	0.24
FDA	200	SGD	0.1	cosine	89.69	89.59	89.56	89.53	89.65	89.60	0.07
	200	SGD	0.1	step	80.38	80.34	80.83	80.52	80.52	80.52	0.19
	200	SGD	0.01	cosine	80.72	80.93	80.89	80.70	80.79	80.81	0.10
	200	SGD	0.01	step	63.41	62.85	63.04	63.04	63.14	63.10	0.20
	200	Adam	0.001	cosine	89.70	89.57	89.57	89.80	89.76	89.68	0.11
	200	Adam	0.001	step	89.84	89.85	90.10	89.79	90.01	89.92	0.13
	200	Adam	0.0001	cosine	89.59	89.10	89.34	89.31	89.51	89.37	0.19
	200	Adam	0.0001	step	89.52	89.59	89.52	89.64	89.58	89.57	0.05

Table 12: Inference Efficiency on Hardware (1/4).

Hardware	Threads	Arch.	Larq			daBNN	
			FP32	BNN	ReAct	FP32	BNN
Kirin 970	1	ResNet18	716.427	123.263	126.457	427.585	72.585
		ResNet34	1449.67	159.615	171.227	836.321	124.091
		VGG-Small	242.443	14.833	16.401	–	–
	2	ResNet18	372.642	72.697	78.605	–	–
		ResNet34	732.355	96.711	108.41	–	–
		VGG-Small	121.91	10.304	11.935	–	–
	4	ResNet18	191.517	42.986	47.182	–	–
		ResNet34	367.891	61.413	73.101	–	–
		VGG-Small	57.721	8.72	8.387	–	–
	8	ResNet18	96.937	37.457	56.017	–	–
		ResNet34	212.982	53.809	67.667	–	–
		VGG-Small	33.647	18.649	19.818	–	–
Kirin 980	1	ResNet18	307.624	49.009	50.018	158.363	31.803
		ResNet34	507.734	71.909	74.920	308.537	53.031
		VGG-Small	83.163	7.772	8.215	–	–
	2	ResNet18	187.494	52.057	54.285	–	–
		ResNet34	367.853	57.336	60.483	–	–
		VGG-Small	49.264	6.116	5.604	–	–
	4	ResNet18	104.076	29.556	35.539	–	–
		ResNet34	202.173	31.324	35.911	–	–
		VGG-Small	22.690	3.147	3.291	–	–
	8	ResNet18	60.307	45.683	56.416	–	–
		ResNet34	120.738	60.758	86.887	–	–
		VGG-Small	18.147	21.688	23.350	–	–
Kirin 985	1	ResNet18	173.238	27.429	30.626	164.556	34.528
		ResNet34	438.971	58.165	60.885	323.439	57.808
		VGG-Small	70.797	6.147	6.796	–	–
	2	ResNet18	103.621	25.672	35.477	–	–
		ResNet34	327.416	53.949	62.865	–	–
		VGG-Small	55.328	5.955	6.243	–	–
	4	ResNet18	92.387	26.728	34.778	–	–
		ResNet34	184.050	39.881	52.153	–	–
		VGG-Small	28.076	8.919	14.795	–	–
	8	ResNet18	130.972	82.772	89.766	–	–
		ResNet34	227.504	119.586	143.958	–	–
		VGG-Small	44.339	34.034	43.816	–	–
Kirin 990	1	ResNet18	114.238	21.235	22.066	144.205	29.239
		ResNet34	227.043	31.545	32.821	275.502	49.476
		VGG-Small	38.118	3.338	3.482	–	–
	2	ResNet18	59.329	13.911	14.179	–	–
		ResNet34	116.822	23.452	22.770	–	–
		VGG-Small	20.055	2.080	2.194	–	–
	4	ResNet18	38.403	10.280	12.208	–	–
		ResNet34	81.273	15.570	17.727	–	–
		VGG-Small	13.508	1.542	1.760	–	–
	8	ResNet18	37.703	25.360	31.365	–	–
		ResNet34	78.753	34.884	39.363	–	–
		VGG-Small	12.707	14.414	21.749	–	–

Table 13: Inference Efficiency on Hardware (2/4).

Hardware	Threads	Arch.	Larq			daBNN	
			FP32	BNN	ReAct	FP32	BNN
Kirin 9000E	1	ResNet18	118.059	19.865	20.547	129.270	24.781
		ResNet34	236.047	31.822	32.575	250.680	42.134
		VGG-Small	39.114	3.595	3.832	–	–
	2	ResNet18	68.351	16.821	16.115	–	–
		ResNet34	133.671	25.061	24.660	–	–
		VGG-Small	23.018	2.684	2.598	–	–
	4	ResNet18	45.592	17.452	18.847	–	–
		ResNet34	91.648	23.395	28.022	–	–
		VGG-Small	14.360	2.762	2.782	–	–
	8	ResNet18	43.363	61.263	42.328	–	–
		ResNet34	89.405	70.232	93.558	–	–
		VGG-Small	19.070	17.351	23.825	–	–
Dimensity 820	1	ResNet18	158.835	32.636	34.912	323.035	63.471
		ResNet34	328.020	57.133	60.807	629.493	102.443
		VGG-Small	82.417	5.958	6.420	–	–
	2	ResNet18	122.167	29.306	34.384	–	–
		ResNet34	250.088	43.306	50.143	–	–
		VGG-Small	51.320	4.670	5.053	–	–
	4	ResNet18	94.636	21.850	30.027	–	–
		ResNet34	177.757	33.809	40.816	–	–
		VGG-Small	45.056	4.223	4.546	–	–
	8	ResNet18	90.210	45.357	61.981	–	–
		ResNet34	166.989	68.444	74.286	–	–
		VGG-Small	32.971	21.344	23.706	–	–
Dimensity 9000	1	ResNet18	106.388	21.023	24.770	148.690	29.030
		ResNet34	210.665	32.841	34.590	284.438	49.854
		VGG-Small	42.057	4.410	5.530	–	–
	2	ResNet18	81.606	22.661	27.050	–	–
		ResNet34	143.349	27.666	37.910	–	–
		VGG-Small	26.512	2.273	2.410	–	–
	4	ResNet18	51.421	13.079	15.200	–	–
		ResNet34	100.249	23.314	25.920	–	–
		VGG-Small	17.735	3.015	3.770	–	–
	8	ResNet18	43.355	24.939	30.740	–	–
		ResNet34	84.182	30.212	39.990	–	–
		VGG-Small	14.857	14.258	17.540	–	–
Snapdragon 855+	1	ResNet18	90.430	19.769	20.530	163.293	31.174
		ResNet34	186.694	29.126	30.512	298.882	49.948
		VGG-Small	29.735	3.153	3.259	–	–
	2	ResNet18	58.510	25.780	26.331	–	–
		ResNet34	124.580	31.023	32.646	–	–
		VGG-Small	19.408	2.258	2.471	–	–
	4	ResNet18	39.269	19.865	23.297	–	–
		ResNet34	82.180	30.248	31.387	–	–
		VGG-Small	13.566	2.032	2.359	–	–
	8	ResNet18	36.630	49.060	85.861	–	–
		ResNet34	73.513	41.131	88.101	–	–
		VGG-Small	12.860	17.828	23.489	–	–

Table 14: Inference Efficiency on Hardware (3/4).

Hardware	Threads	Arch.	Larq			daBNN	
			FP32	BNN	ReAct	FP32	BNN
Snapdragon 870	1	ResNet18	88.145	16.527	17.020	126.762	25.240
		ResNet34	185.468	25.488	26.195	237.361	41.440
		VGG-Small	30.318	2.851	2.964	–	–
	2	ResNet18	63.829	18.351	19.575	–	–
		ResNet34	159.174	25.352	26.340	–	–
		VGG-Small	27.669	2.094	2.308	–	–
	4	ResNet18	42.796	17.578	21.083	–	–
		ResNet34	89.960	25.816	27.201	–	–
		VGG-Small	14.829	2.614	2.215	–	–
	8	ResNet18	46.798	19.192	28.579	–	–
		ResNet34	97.834	25.060	32.863	–	–
		VGG-Small	16.799	9.717	17.293	–	–
Snapdragon 888	1	ResNet18	77.205	15.547	16.111	123.618	25.240
		ResNet34	152.887	22.906	23.893	234.972	41.648
		VGG-Small	25.133	2.410	2.543	–	–
	2	ResNet18	46.297	19.309	19.321	–	–
		ResNet34	93.615	20.473	22.489	–	–
		VGG-Small	16.001	1.920	2.213	–	–
	4	ResNet18	33.524	13.699	14.332	–	–
		ResNet34	67.495	19.020	21.157	–	–
		VGG-Small	11.743	2.882	2.768	–	–
	8	ResNet18	33.761	26.108	58.989	–	–
		ResNet34	67.876	37.018	61.315	–	–
		VGG-Small	11.752	27.615	16.774	–	–
Raspberrypi 3B+	1	ResNet18	740.509	158.732	175.256	1460.723	241.713
		ResNet34	1536.915	240.606	254.810	2774.888	435.170
		VGG-Small	257.079	24.479	25.790	–	–
	2	ResNet18	667.012	143.716	106.894	–	–
		ResNet34	933.149	144.287	158.868	–	–
		VGG-Small	145.427	14.503	15.628	–	–
	4	ResNet18	562.567	108.585	116.640	–	–
		ResNet34	976.223	159.258	183.698	–	–
		VGG-Small	191.470	10.839	10.196	–	–
	8	ResNet18	877.026	279.660	356.239	–	–
		ResNet34	1638.035	389.924	485.260	–	–
		VGG-Small	399.338	110.448	142.978	–	–
Raspberrypi 4B	1	ResNet18	448.744	80.822	82.380	688.838	120.348
		ResNet34	897.735	112.837	119.536	1362.893	209.276
		VGG-Small	150.814	11.177	12.024	–	–
	2	ResNet18	261.861	49.079	55.279	–	–
		ResNet34	525.735	67.480	79.468	–	–
		VGG-Small	89.284	6.647	7.882	–	–
	4	ResNet18	270.191	36.331	45.903	–	–
		ResNet34	572.423	53.866	70.841	–	–
		VGG-Small	90.650	5.056	6.167	–	–
	8	ResNet18	466.585	168.844	226.771	–	–
		ResNet34	879.375	264.638	319.789	–	–
		VGG-Small	216.439	114.064	162.118	–	–

Table 15: Inference Efficiency on Hardware (4/4).

Hardware	Threads	Arch.	Larq			daBNN	
			FP32	BNN	ReAct	FP32	BNN
Apple M1	1	ResNet18	44.334	8.219	8.355	–	–
		ResNet34	88.334	12.505	12.771	–	–
		VGG-Small	14.093	1.446	1.465	–	–
	2	ResNet18	24.775	5.037	5.194	–	–
		ResNet34	47.179	7.425	7.690	–	–
		VGG-Small	7.398	0.829	0.854	–	–
	4	ResNet18	18.612	3.448	3.671	–	–
		ResNet34	27.515	4.965	5.254	–	–
		VGG-Small	4.294	0.526	0.551	–	–
	8	ResNet18	16.653	5.035	6.003	–	–
		ResNet34	27.680	6.445	6.953	–	–
		VGG-Small	3.996	0.735	0.712	–	–
	16	ResNet18	90.323	70.697	73.729	–	–
		ResNet34	162.057	130.907	125.362	–	–
		VGG-Small	25.366	23.050	23.194	–	–
Apple M1 Max	1	ResNet18	46.053	8.653	8.486	–	–
		ResNet34	91.861	12.593	13.039	–	–
		VGG-Small	14.285	1.454	1.488	–	–
	2	ResNet18	25.039	5.450	5.361	–	–
		ResNet34	51.860	7.579	8.925	–	–
		VGG-Small	7.657	0.855	0.896	–	–
	4	ResNet18	14.708	3.625	3.888	–	–
		ResNet34	27.933	5.266	6.021	–	–
		VGG-Small	4.292	0.576	0.620	–	–
	8	ResNet18	10.660	3.718	4.510	–	–
		ResNet34	18.988	4.745	5.457	–	–
		VGG-Small	3.432	0.560	0.629	–	–
	16	ResNet18	60.500	47.727	53.900	–	–
		ResNet34	120.449	91.464	96.356	–	–
		VGG-Small	21.354	13.868	15.311	–	–