QuartDepth: Post-Training Quantization for Real-Time Depth Estimation on the Edge

Xuan Shen[1], Weize Ma[2], Jing Liu[3], Changdi Yang[1], Rui Ding[2], Quanyi Wang[4], Henghui Ding[5], Wei Niu[6], Yanzhi Wang[1], Pu Zhao[1]*, Jun Lin[2]*, Jiuxiang Gu[7]*
[1]Northeastern University, [2]Nanjing University, [3]Monash University,
[4]Nanjing University of Information Science and Technology, [5]Fudan University,
[6]University of Georgia, [7]Adobe Research
{shen.xu,yanz.wang,p.zhao}@northeastern.edu, weizema@smail.nju.edu.cn,
jlin@nju.edu.cn, jigu@adobe.com

Abstract

Monocular Depth Estimation (MDE) has emerged as a pivotal task in computer vision, supporting numerous real-world applications. However, deploying accurate depth estimation models on resource-limited edge devices, especially Application-Specific Integrated Circuits (ASICs), is challenging due to the high computational and memory demands. Recent advancements in foundational depth estimation deliver impressive results but further amplify the difficulty of deployment on ASICs. To address this, we propose ***QuartDepth*** *which adopts post-training quantization to quantize MDE models with hardware accelerations for ASICs. Our approach involves quantizing both weights and activations to 4-bit precision, reducing the model size and computation cost. To mitigate the performance degradation, we introduce activation polishing and compensation algorithm applied before and after activation quantization, as well as a weight reconstruction method for minimizing errors in weight quantization. Furthermore, we design a flexible and programmable hardware accelerator by supporting kernel fusion and customized instruction programmability, enhancing throughput and efficiency. Experimental results demonstrate that our framework achieves competitive accuracy while enabling fast inference and higher energy efficiency on ASICs, bridging the gap between high-performance depth estimation and practical edge-device applicability. Code:* https://github.com/shawnricecake/quart-depth

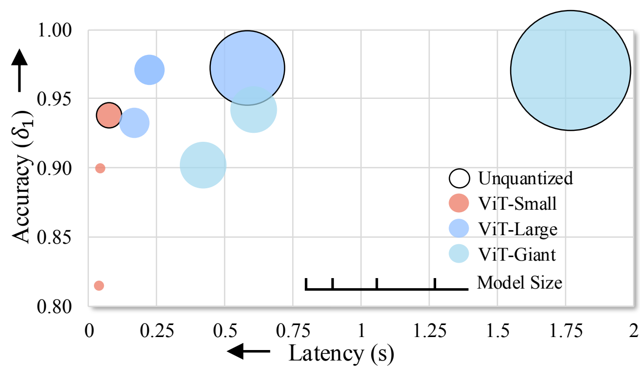

Figure 1. Visualization of accuracy vs. latency on ASIC for monocular depth estimation models with different backbones.

1. Introduction

Monocular Depth Estimation (MDE) is a critical task in computer vision, essential for a wide range of applications including robotics [82], autonomous driving [77, 97], virtual reality [57], and 3D reconstruction [54, 92, 105]. The objective of MDE is to estimate depth information from a single image, making it particularly valuable for scenarios where stereo or multi-view depth sensors are impractical. Previously, depth estimation methods have been classified into three main approaches: learning metric depth [3, 89, 91, 99], learning relative depth [8, 9, 86, 87], and learning affine-invariant depth [54, 56, 93, 94, 105]. Among these, metric depth methods have demonstrated remarkable performance across benchmarks but often show limited generalizability to diverse, real-world images, which is compounded by variations in camera parameters between training and test sets.

Recently, inspired by the rise of "foundation models" [5, 107], a new wave of MDE models [4, 66, 90, 95] has been developed. These foundational MDE models exhibit superior generalization and robustness, effectively extending their applicability to in-the-wild images and accommodating a wide array of camera parameters. However, while these foundational MDE models achieve impressive performance,

*Corresponding Author

they rely on sophisticated and computationally intensive algorithms. This reliance on complex architectures and operations significantly increases computational costs, making such models resource-heavy and challenging to deploy on edge devices or specialized hardware such as Application-Specific Integrated Circuits (ASICs) [35, 52, 81, 82]. The high computational demands also hinder real-time performance and scalability, which are critical for practical applications such as autonomous navigation [2], robotic perception [80], and augmented reality [68]. This constraint underscores the need for more efficient approaches that can maintain high accuracy while reducing computational overhead. Among common techniques like pruning [16, 64, 82, 106], knowledge distillation [18, 50], and architecture optimization [1, 37, 38, 63, 85, 101], which mainly reduce model complexity, quantization [11, 36, 46, 79] stands out for achieving both compactness and speed. Quantization [14, 39, 41, 84, 100] reduces weights and activations to lower bit-widths (e.g., 4-bit or 8-bit integers), minimizing model size while accelerating inference through optimized integer kernels specifically designed for edge devices. This dual benefit makes it ideal for deploying high-performing models on resource-limited edge hardware. Given the complexity of large foundational MDE models and their extensive datasets [31, 34, 98], we employ Post-Training Quantization (PTQ) [36, 41, 62, 100, 108] to facilitate efficient deployment without retraining.

In this paper, we propose **QuartDepth**, a post-training quantization framework designed for real-time depth estimation on edge devices. We begin by analyzing the outlier deviant distribution in MDE foundation models [90, 95]. Through an in-depth per-channel analysis of these abnormal distributions, we identify persistent extreme outliers in the depth decoders. To address this, we propose the LogNP polishing optimization method to smooth the outliers and transform the abnormal distribution into a more quantization-friendly, normalized distribution. Furthermore, to mitigate the loss from activation quantization, we employ a compensation algorithm that updates the weights to minimize the activation quantization error. Next we perform weight quantization with a weight reconstruction method that leverages gradients to minimize second-order weight quantization errors. Meanwhile, we design a novel flexible and programmable hardware accelerator by supporting kernel fusion and customized instruction programmability, which enables direct processing of intermediate results and concurrent execution of computational tasks. In detail, we design specialized computation kernels for W4A4 and W4A8 configurations, which enables full utilization of external memory bandwidth. Also, we design a novel programmable vector computation array to support the proposed LogNP polishing optimization, effectively hiding its additional overhead.

In summary, our contributions are outlined as follows,

1. We observe a challenging outlier deviant distribution for activations and propose the LogNP polishing for activation quantization, which transforms these outliers into a quantization-friendly, normalized distribution.
2. For weight quantization, we first update the weights to compensate the error of activation quantization, and then quantize the updated weights with weight reconstruction to minimize the second-order weight quantization error.
3. We develop a novel flexible and programmable hardware accelerator on ASICs corresponding to our proposed quantization methods.
4. Comprehensive experiments confirm the effectiveness of our QuartDepth framework with superior accuracy, real-time inference and higher power efficiency on ASICs.

2. Related Work

2.1. Efficient Depth Estimation

Depth estimation from a single color image has been a prominent research focus in the field of computer vision for over a decade. This task plays a critical role in various applications, including robotic perception [12, 13, 82], autonomous driving [77, 97], virtual reality [57], and 3D reconstruction [54, 92, 105], all of which require rapid and accurate depth perception. The demand for real-time processing in these applications highlights the necessity for efficient model deployment. To meet these requirements, advanced optimization techniques such as pruning [16, 42, 65, 82, 104], knowledge distillation [18, 50], and architecture optimization [1, 88, 108] have been adopted to compress and accelerate the model. However, previous research has primarily focused on relatively smaller models, which, while effective, do not match the scale and capability of foundational MDE models [90, 95] . These modern foundational MDE models [4, 66] have gained significant attention for their ability to achieve state-of-the-art performance and generalize effectively across diverse real-world scenarios. By leveraging massive datasets and advanced architectures, these models provide robust depth estimation at the cost of increased computational complexity and larger model sizes. Thus, there is an urgent need for new model compression and acceleration techniques specifically designed to meet real-time requirements on resource-limited edge devices.

2.2. Hardware Design

Deploying large foundational models on edge devices presents significant challenges due to stringent constraints on computation and memory access [69]. To tackle these issues, various optimization techniques [59–62, 83, 102, 103, 108] have been explored to reduce model size and computational overhead. However, models subjected to such aggressive optimizations often fail to achieve peak performance on general-purpose CPUs and GPUs [96]. In contrast, ASICs,

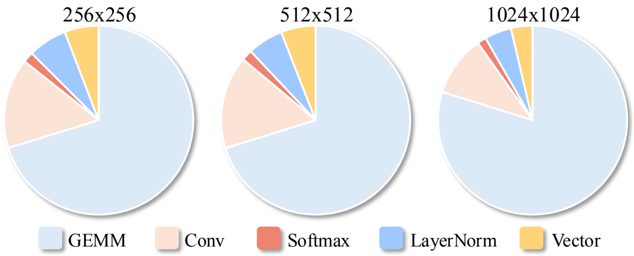

Figure 2. Latency profiling for 3 different resolutions using Metric3D with ViT-Large backbone. Vector includes activation functions (e.g. ReLU) and element-wise addition and multiplication.

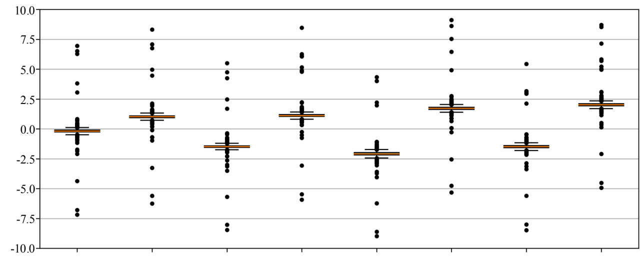

Figure 3. Visualization of outliers in various channels at the decoder [55] of Metric3D model on NYUv2 dataset.

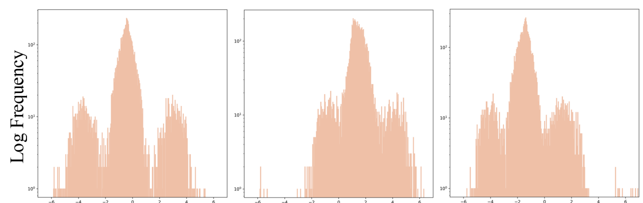

Figure 4. Visualization for the activation data distribution of different channels in decoder with log frequency.

optimized for specific models and algorithms, offer significant performance and energy efficiency gains, making them ideal for deploying large foundational models on edge devices [22, 23, 43, 53, 78, 96]. Early work, such as TPU [28], DNA [72], and Eyeriss [10], significantly improved inference speed and energy efficiency by focusing on specialized hardware designs tailored to the needs of deep learning models. Recent works [29, 30, 49, 73, 74] have proposed dedicated hardware architectures in ASICs to accelerate neural network inference. These ASIC architectures have significantly improved inference latency and energy efficiency through solutions such as quantization, sparsity, data flow reconstruction, and algorithm-hardware co-design [40]. Thus, to maximize potential of large foundational models, the advantages offered by ASICs make them platform of choice.

3. Analysis and Motivation

3.1. Latency Profiling

To optimize edge device inference, we first profile the Float32 model to identify the most time-consuming components for targeted optimization. In detail, we implement the Metric3D [95] model with ViT-Large [15] backbone on a hardware performance emulator with our ASIC design. The latency profiling results in Figure 2 show that the matrix multiplication and convolution operations dominate the inference time, while non-linear operations like softmax and layer normalization contribute minimally. The vector operations include activation functions such as SiLU, ReLU, and GeLU, as well as element-wise addition and multiplication, which also contribute minimally to the overall latency. This motivates our focus on quantization and ASIC design for linear and convolution layers, keeping non-linear operations in floating point to preserve accuracy.

3.2. Deviant Distribution

We investigate the deviant distribution in MDE models from a per-channel perspective. In Figure 3, we visualize the outliers across various channels of an activation in the decoder [55] of the Metric3D model on NYUv2 dataset. We observe large amount of outliers and significant variability in outlier values across different channels, making per-tensor quantization challenging. This motivates the adoption of a per-channel quantization method to better handle these variations. However, for per-channel quantization, the presence of outliers that deviate significantly from the main data distribution, as shown in Figure 4, still leads to substantial quantization errors. Thus, smoothing these large outliers and integrating them into the main data distribution is crucial for maintaining model accuracy.

4. Quantization Framework

We first quantize the polished activations with LogNP polishing (Section 4.2). Then we update weights to compensate the loss of activation quantization (Section 4.3). Finally, we quantize updated weights (Section 4.4).

4.1. Preliminary

Quantization. We mainly adopt the hardware-efficient PTQ methods. For uniform quantization, the quantize and dequantize operations for x can be defined as follows,

$$x_q = \text{CLIP}\left(\left\lfloor \frac{x}{s} \right\rceil + zp, 0, 2^k - 1\right), \tag{1}$$

$$\hat{x} = s \cdot (x_q - z), \tag{2}$$

where s represents the scale and zp is the zero point. The operator $\lfloor \cdot \rceil$ indicates rounding to the nearest integer, and the CLIP function constrains values that fall outside the range of a k-bit integer.

As for Log2 quantization, which is mainly used for post-softmax quantization, can be defined as follows,

$$x_q = \text{CLIP}\left(\left\lfloor -\log_2 \frac{x}{s} \right\rceil, 0, 2^k - 1\right), \tag{3}$$

$$\hat{x} = s \cdot 2^{-x_q}. \tag{4}$$

Notations. The quantized layers (including both linear and convolutional layers) in the mode are indexed by $l, 1 \leq l \leq L$. The weights in the l^{th} layer are denoted by $\mathbf{W}^{(l)}$ with its vectorized/flattened version $\mathbf{w}^{(l)}$. $\mathbf{x}$ denotes its inputs.

4.2. Activation Polishing

Building on analysis in Section 3.2, we propose LogNP polishing method to adaptively smooth the deviant distribution in decoders in per-channel prospective. For one activation $\mathbf{x} \in \mathbb{R}^{n \times d}$ where n denotes the number of tokens and d denotes the channel number, the LogNP polishing function $\Phi(\cdot, \cdot)$ for each element x of input $\mathbf{x}$ is defined as follows,

$$\Phi(x, \alpha) = \text{sign}(x) \cdot \left[\log_2\left(|x| + \alpha\right) - \log_2(\alpha)\right], \tag{5}$$

where sign$(\cdot)$ denotes the sign function, and α is the polishing factor for one single channel.

To reduce the quantization error in each channel, the polishing factor α_i corresponding to the i^{th} channel with activation $\mathbf{x}_i$ (i.e., $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_d]$) is computed according to the given percentile ϵ as follows,

$$\alpha_i = P_\epsilon(\mathbf{x}_i),\ i \in \{1, 2, 3, ..., d\},\ \epsilon \in (0, 100) \tag{6}$$

where $P_\epsilon(\cdot)$ denotes the ϵ-th percentile function. We adopt uniform ϵ equals to 95 as deviant activation distributions remain similar across different inputs. We provide the visualization of the polished activation in Figure 5. The polished activation colored in blue appears smoother compared to the original activation colored in red shown in Figure 4, making it more friendly for quantization. After polishing, we perform activation quantization with uniform quantization in Section 4.1. Following activation dequantization, we adopt reverse transformation (i.e., unpolishing) as follows,

$$\Phi^{-1}(\hat{x}, \alpha) = \text{sign}(\hat{x}) \cdot \left[2^{\text{sign}(\hat{x})\cdot\hat{x} + \log_2(\alpha)} - \alpha\right] \tag{7}$$

The latency overhead introduced by LogNP polishing and unpolishing is hidden by concurrent execution in our hardware design, which is further explained in Section 5.2.

4.3. Activation Loss Compensation

The activation quantization introduces quantization errors. To mitigate this issue, before weight quantization, we first update all weights to compensate the loss of activation quantization. To save the computation cost, we address the problem

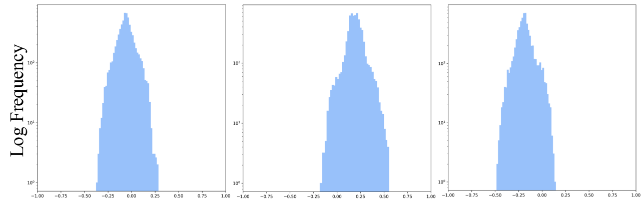

Figure 5. Visualization for the polished activation data distribution of different channels in decoder with log frequency.

layer by layer instead of for the whole model. The layer-wise compensation problem is formulated as the following,

$$\min_{\Delta \mathbf{W}} \|\mathbf{W}\mathbf{x} - (\mathbf{W} + \Delta\mathbf{W})\hat{\mathbf{x}}\|_2^2 \tag{8}$$

where $\mathbf{W}$ and $\mathbf{x}$ are the original weights and inputs of the layer. $\hat{\mathbf{x}}$ is the quantized inputs and we update the weights with $\Delta\mathbf{W}$ to compensate the activation quantization error, so that the layer outputs after activation quantization and weight compensation should be close to the original outputs.

This problem can be solved by setting the gradients of the minimization objective to zero, as the following,

$$\Delta\mathbf{W}\hat{\mathbf{x}}\hat{\mathbf{x}}^{\mathrm{T}} + \mathbf{W}(\mathbf{x} - \hat{\mathbf{x}})\hat{\mathbf{x}}^{\mathrm{T}} = 0. \tag{9}$$

The optimal solution can be obtained by

$$\Delta\mathbf{W}^* = -\mathbf{W}(\mathbf{x} - \hat{\mathbf{x}})\hat{\mathbf{x}}^{\mathrm{T}}(\hat{\mathbf{x}}\hat{\mathbf{x}}^{\mathrm{T}})^{-1} \tag{10}$$

After activation quantization, we can update weights with the above $\Delta\mathbf{W}^*$ to compensate the activation quantization error. If $\hat{\mathbf{x}}\hat{\mathbf{x}}^{\mathrm{T}}$ is not full rank with difficulties for matrix inversion, we adopt the dampening technique [21, 45]. Next, we quantize the updated weights under the quantized activations.

4.4. Weight Reconstruction

Problem formulation. In quantization, the weights are modified and we would like to minimize the change of loss incurred by quantization to maintain the superior performance of the model. To quantitatively analyze the loss degradation by quantization, we approximate the loss degradation with Taylor expansion as the following,

$$\begin{aligned} L(\mathbf{w} + \Delta\mathbf{w}) - L(\mathbf{w}) &\approx \Delta\mathbf{w}^{\mathrm{T}}\mathbf{g}_{\mathbf{w}} + \frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\mathbf{H}_{\mathbf{w}}\Delta\mathbf{w}, \\ &\approx \frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\mathbf{H}_{\mathbf{w}}\Delta\mathbf{w}, \end{aligned} \tag{11}$$

where $\mathbf{g}_{\mathbf{w}} = \nabla_{\mathbf{w}} L$ represents the gradients and $\mathbf{H}_{\mathbf{w}} = \nabla_{\mathbf{w}}^2 L$ denotes the Hessian matrix. $\Delta\mathbf{w}$ is the weight perturbation in this step. Here we use the vectorized/flattened version $\mathbf{w}$ instead of the matrix version $\mathbf{W}$ to obtain a scalar output. As the pre-trained model has been well trained and converged to a minimum, it is reasonable to assume the gradients to be $\mathbf{0}$. The complexity to obtain the full Hessian is very high and typically it is not possible to obtain the full Hessian due to computation and memory limitations. Specifically, $\mathbf{H_w} \in \mathbb{R}^{B\times B}$ where B is the number of parameters in $\mathbf{w}$. The memory cost for $\mathbf{H_w}$ is extremely heavy in LLMs with billions of parameters.

Approximating Hessian. For a network trained with negative log-likelihood loss and a probabilistic model $p_{\mathbf{w}}(y|x_s)$ under softmax, the Hessian is identical to the Fisher matrix:

$$\mathbf{H_w} = \mathbf{F_w} = \sum_{s=1}^{S} \mathbb{E}_{y\sim p_{\mathbf{w}}(y|x_s)}[\nabla_{\mathbf{w}} \log p_{\mathbf{w}}(y|x_s) \nabla_{\mathbf{w}} \log p_{\mathbf{w}}(y|x_s)^T] \quad (12)$$

where S is the number of samples, and for each sample, it needs to compute the expectation over all categories. The complexity with the expectation is also high if there are too many categories. Thus, in practice, the *empirical Fisher* is commonly adopted to replace the expectation over y with target label y_s [33]. The dimension of $\mathbf{F_w}$ is the same as $\mathbf{H_w}$ with quadratic memory complexity to the number of parameters. To reduce the complexity, the empirical Fisher is computed by treating layers independently and the layer-wise empirical Fisher can be obtained as below [6, 44],

$$\mathbf{F}_l = \mathbb{E}\left[(\mathbf{g}^{(l)}\mathbf{g}^{(l),\mathrm{T}}) \otimes (\hat{\mathbf{x}}^{(l)}\hat{\mathbf{x}}^{(l),\mathrm{T}})\right], \quad (13)$$

where $\hat{\mathbf{x}}^{(l)}$ is the input activations and $\mathbf{g}^{(l)}$ is the gradients of the l^{th} layer. $\otimes$ denotes Kronecker product. The expectation is taken with respect to data distribution over inputs.

For two different layers (for example, the i^{th} and the j^{th} layers), their Fisher is often written as $\mathbf{F}_{ij} = \sum_{s=1}^{S} \mathbb{E}\left[\nabla_{\mathbf{w}^{(i)}} \log p_{\mathbf{w}}(y_s|x_s) \nabla_{\mathbf{w}^{(j)}} \log p_{\mathbf{w}}(y_s|x_s)^T\right]$, which are often ignored to save computations. The Fisher for the whole model can be formulated as a block diagonal matrix $\mathbf{F_w} = \mathrm{diag}(\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_L)$.

To obtain the Fisher in practice, we adopt the KFAC approximation method [20, 44] as below,

$$\mathbf{F}_l = \mathbf{G}_l \otimes \mathbf{A}_l \quad (14)$$

$$\mathbf{G}_l = \frac{1}{\sqrt{S}} \sum_{s=1}^{S} \mathbf{g}_s^{(l)} \mathbf{g}_s^{(l),\mathrm{T}} \quad (15)$$

$$\mathbf{A}_l = \frac{1}{\sqrt{S}} \sum_{s=1}^{S} \hat{\mathbf{x}}_s^{(l)} \hat{\mathbf{x}}_s^{(l),\mathrm{T}} \quad (16)$$

The KFAC approximation [25, 76] approximates an expectation of Kronecker products as a Kronecker product of two expectations $\mathbb{E}[\mathbf{g}^{(l)}\mathbf{g}^{(l),T} \otimes \hat{\mathbf{x}}^{(l)}\hat{\mathbf{x}}^{(l),T}] \approx \mathbb{E}[\mathbf{g}^{(l)}\mathbf{g}^{(l),T}] \otimes \mathbb{E}[\hat{\mathbf{x}}^{(l)}\hat{\mathbf{x}}^{(l),T}]$, where the activations and derivatives are assumed to be independent.

Optimizing quantization parameters. Next we optimize the quantization parameters. Given our optimization loss in Equation (11), multiple optimization methods can be adopted such as STE [24] and AdaRound [46]. Due to its superior performance in PTQ, we use AdaRound [46] to learn the rounding parameter which decides rounding up or down. Formally, we minimize the following loss,

$$\min_{\mathbf{v}} \sum_{l=1}^{L} \left(\mathbf{w}^{(l)} - \hat{\mathbf{w}}^{(l)}\right)^{\mathrm{T}} \mathbf{F}_l \left(\mathbf{w}^{(l)} - \hat{\mathbf{w}}^{(l)}\right) + \lambda h(\mathbf{v}), \quad (17)$$

where $\mathbf{v}$ denotes the rounding parameter which are used to construct the quantized weights $\hat{\mathbf{w}}^{(l)}$. $h(\mathbf{v})$ is a regularizer in AdaRound [46] with λ controlling its strength. More details about AdaRound are presented in Appendix 9. We only use a small calibration set (32 samples) sampled from the training dataset to calibrate the quantized model.

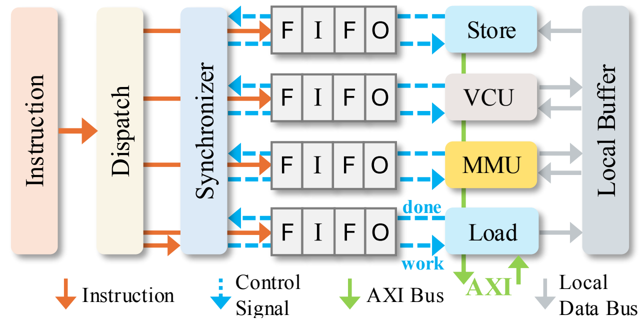

Figure 6. The block diagram of the proposed hardware accelerator.

5. Hardware Design

5.1. Top-Level Design

We design a flexible and programmable accelerator as shown in Figure 6. The Dispatch module fetches instructions via the AXI bus and distributes them to local Instruction FIFOs, while the Synchronize module controls the start and end of each module. Instruction FIFOs hold out-of-order instructions and are controlled by Synchronize module for execution. The Load and Store modules, functioning as the DMA, facilitate data transfer between off-chip memory (e.g., HBM or DDR) and local buffers. The Matrix Multiplication Unit (MMU) performs General Matrix Multiply (GeMM) and convolution when the Vector Compute Unit (VCU) handles nonlinear activation functions.

5.2. On-Chip Kernel Design

We adopt a multiply-accumulate tree array to implement both matrix multiplication and convolution. For W4A4 and W4A8 configurations, we design specialized multipliers and adders to enhance computational efficiency and reduce chip area overhead. Additionally, the use of low-bit quantized matrices eases memory bandwidth demands during data transfer, allowing for full utilization of the external memory bandwidth. As for Float32, we implement IEEE 754 standard floating-point computation for fair comparison.

Meanwhile, we propose a programmable vector computation array composed of multiple Floating-Point Units (FPU)

Method	W / A	NYUv2 [47]						KITTI [19]					
		AbsRel ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	RMSE ↓	Silog ↓	AbsRel ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	RMSE ↓	Silog ↓
ViT-Small Backbone													
\	Float32	0.087	0.938	0.990	0.996	0.331	0.035	0.074	0.934	0.984	0.995	3.403	0.043
OBS [17]	W4	0.107	0.885	0.985	0.997	0.391	0.042	0.122	0.867	0.973	0.991	4.508	0.058
minmax [27]	W4A8	0.453	0.405	0.702	0.868	1.067	0.111	0.456	0.102	0.329	0.575	13.355	0.159
ema [26]		0.257	0.591	0.859	0.958	0.793	0.093	0.312	0.341	0.632	0.807	11.102	0.139
percentile [11]		0.190	0.635	0.912	0.981	0.774	0.077	0.222	0.552	0.814	0.918	8.900	0.114
AdaRound [46]		0.134	0.793	0.967	0.981	0.471	0.042	0.128	0.876	0.953	0.993	4.236	0.051
BrecQ [36]		0.128	0.817	0.980	0.996	0.452	0.040	0.104	0.900	0.977	0.993	3.914	0.047
Ours		**0.103**	**0.899**	**0.987**	**0.996**	**0.380**	0.042	**0.092**	**0.916**	**0.981**	**0.995**	**3.682**	**0.046**
minmax [27]	W4A4	1.265	0.130	0.285	0.477	2.593	0.160	0.584	0.054	0.160	0.321	15.877	0.245
ema [26]		0.371	0.359	0.359	0.841	1.226	0.146	0.632	0.022	0.081	0.228	16.265	0.244
percentile [11]		0.357	0.366	0.659	0.846	1.223	0.146	0.646	0.019	0.064	0.197	16.384	0.244
AdaRound [46]		0.251	0.459	0.812	0.936	0.848	0.103	0.501	0.239	0.368	0.538	14.398	0.216
BrecQ [36]		0.217	0.598	0.884	0.970	0.724	0.071	0.405	0.278	0.493	0.634	12.337	0.184
Ours		**0.133**	**0.815**	**0.965**	**0.990**	**0.453**	**0.052**	**0.197**	**0.534**	**0.937**	**0.985**	**5.429**	**0.059**
ViT-Large Backbone													
\	Float32	0.067	0.972	0.993	0.997	0.262	0.040	0.054	0.974	0.995	0.999	2.505	0.032
OBS [17]	W4	0.070	0.972	0.994	0.998	0.267	0.038	0.060	0.968	0.995	0.998	2.748	0.034
minmax [27]	W4A8	0.671	0.279	0.547	0.748	1.448	0.144	0.546	0.055	0.172	0.385	14.841	0.202
ema [26]		0.657	0.289	0.559	0.756	1.425	0.141	0.394	0.219	0.481	0.686	12.085	0.169
percentile [11]		0.425	0.349	0.715	0.907	1.073	0.106	0.192	0.645	0.883	0.949	6.957	0.093
AdaRound [46]		0.084	0.959	0.991	0.997	0.276	0.040	0.057	0.972	0.995	0.999	2.769	0.034
BrecQ [36]		0.076	0.967	0.993	0.997	0.272	0.039	0.057	0.971	0.995	0.999	2.778	0.034
Ours		**0.071**	**0.970**	**0.993**	**0.997**	**0.264**	**0.036**	**0.055**	**0.973**	**0.995**	**0.999**	**2.542**	**0.032**
minmax [27]	W4A4	2.238	0.116	0.249	0.391	5.761	0.261	0.565	0.063	0.190	0.363	15.693	0.253
ema [26]		1.039	0.174	0.376	0.578	2.126	0.157	0.508	0.123	0.294	0.460	15.206	0.247
percentile [11]		0.920	0.207	0.428	0.631	1.915	0.157	0.492	0.144	0.317	0.487	15.012	0.244
AdaRound [46]		0.501	0.259	0.564	0.782	1.482	0.145	0.482	0.172	0.395	0.541	13.782	0.207
BrecQ [36]		0.469	0.330	0.609	0.811	1.323	0.137	0.434	0.191	0.407	0.590	13.462	0.192
Ours		**0.097**	**0.932**	**0.993**	**0.998**	**0.327**	**0.045**	**0.070**	**0.952**	**0.993**	**0.998**	**3.104**	**0.038**

Table 1. Main results of Metric3D model with ViT-Small and ViT-Large backbone on NYUv2 and KITTI datasets. Results with ViT-Giant backone is included in Table A1 at Appendix 10.

in Vector Computing Unit (VCU). In the VCU, we design a programmable vector computation array composed of multiple FPU to enable more efficient computation. The FPU supports the Special Function Unit (SFU) through polynomial approximation [51], which achieves the computation of non-linear functions (e.g. $\log_2$ and $\exp$) within 3 to 5 clock cycles while maintaining ultra-high precision. For the activation polishing optimization discussed in Section 4.2, the FPU is further specifically designed to enhance execution efficiency. Compared to the NVIDIA GPU [48], the proposed SFU achieves nearly same accuracy, with only a 2 to 5 ULP (units in the last place) error. Meanwhile, the VCU supports multiple numerical format inputs, unifying them into Float32 for computation, and then converts these Float32 numbers into integers such as INT4 or INT8 as needed, which seamlessly integrates both floating-point and integer data types for efficient processing. Hence, the proposed quantization framework for MDE models with activation polishing optimization is accurately and efficiently deployed on our hardware.

5.3. Overlapped Computation Flow

Our hardware architecture supports kernel fusion of on-chip matrix multiplication and vector computation, enabling intermediate results stored in local SRAM to be directly sent to the VCU for activation, quantization, or dequantization without being written back to DDR. By fusing these operations into a single flow, we minimize the latency introduced by multiple memory accesses. Also, this design reduces data transfer time and enhances the overall execution efficiency. Furthermore, our hardware is designed to be programmable with customized instructions, enabling the configuration of different computational tasks to be executed concurrently within an instruction cycle. This parallelism enables a substantial speedup in processing by allowing multiple operations to be pipelined, thereby reducing the total time needed to complete complex tasks. As shown in Figure 7, data transferring, matrix computation, and vector computation can be fully overlapped to improve execution efficiency. This concurrency ensures that no computational resource is idle and every cycle is utilized to its maximum potential, leading to a streamlined and optimized workflow. Therefore, the total on-device latency of the depth estimation model executed with our quantization framework is optimized through extensive operator fusion and concurrent execution.

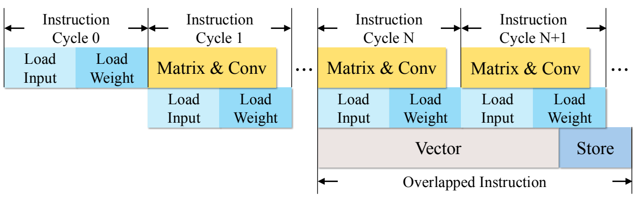

Figure 7. The schematic of concurrent execution, where the computation of Matrix, Vector, Load and Store can execute parallel.

6. Experimental Results

6.1. Experiment Setup

We adopt Metric3D [95] with ViT-Small, ViT-Large, and ViT-Giant [15] backbones to present the main results and Depth Anything [90] with ViT-Large backbone to further verify the generalization across additional datasets. For a fair comparison, we adopt per-channel asymmetric quantization for both weights and activations in W4A8 and W4A4 configurations. We randomly sample 32 images as calibration set from the training sets of NYUv2 [47] for indoor scenes and KITTI [19] for outdoor scenes. The percentile ϵ for the polishing factor is set to 95, and α is then computed as the average value across the calibration samples. For activation loss compensation and weight reconstruction, we still use same 32 samples for the calibration. For gradient computation in weight reconstruction, we adopt batch size of 1, learning rate set to 4e-5, warm up of 0.2, weight decay of 0.01, drop rate of 0.5, and 20,000 iterations. For evaluation, we use NYUv2 [47], SUN RGB-D [67], iBims-1 [32], and HyperSim [58] for indoor scenes; and KITTI [19], vKITTI [7], and DIODE [75] for outdoor scenes. Absolute relative error (AbsRel), accuracy under threshold ($\delta_i < 1.25^i, i = 1, 2, 3$), root mean squared error (RMSE), and metric-depth loss (Silog) metrics are employed in our evaluation results.

6.2. Hardware Implementation

We implement our hardware in RTL and synthesis the design using Design Compiler [70] under a commercial 28nm CMOS technology. The frequency and area are tested by the Design Compiler and the power is tested by Prime Time [71] PX with synthesized netlist and dynamic simulation switch rate under different modes. After synthesis, based on the latency profiled by simulation and the frequency given by Design Compiler, we introduce a hardware performance emulator for latency test. The DDR is simulated by RTL, with a bandwidth of 19.2 GB/s.

6.3. Depth Estimation Results

We present the quantization results with W4A8 and W4A4 configurations using Metric3D [95] with ViT-Small and ViT-Large backbone on indoor dataset NYUv2 [47] and outdoor dataset KITTI [19] in Table 1. The results with ViT-Giant backbone are included in Table A1 at Appendix 10. Compared to conventional PTQ methods such as minmax [27], EMA [26], and percentile [11], our framework demonstrates significantly improved performance. Additionally, when benchmarked against other learning-based approaches including OBS [17], AdaRound [46], and BrecQ [36], our method also achieves superior results. Notably, for the ViT-Large backbone with W4A8 settings, our method maintains nearly lossless performance. In the W4A4 configuration, our approach shows a substantial performance advantage over other methods, validating the effectiveness of our proposed quantization framework. Meanwhile, we present results of Depth Anything model with ViT-Large backbone on various additional datasets [7, 32, 58, 67, 75] covering both indoor and outdoor scenes, as shown in Table 2. Our method, which applies quantization to both weights and activations, achieves performance comparable to OBS (a weight-only quantization method) across multiple datasets in W4A8 configuration, and outperforms the BrecQ for all configurations, showing the effectiveness of our proposed quantization method.

6.4. Latency Results on ASIC

We present the latency results of Metric3D with detailed hardware resource utility in Table 3. The latency is tested with three different resolutions. The power is further normalized by the latency and computation cost (i.e., GMACs) for the energy efficiency. For three different configurations, the frequency are uniformly 1 GHz. The silicon area that our design occupies on a chip for Float32, W4A8, and W4A4 is 29.22 mm^2, 23.94 mm^2, and 24.35 mm^2, respectively. As shown in Table 3, our proposed method demonstrates superior performance with faster inference speed and higher power efficiency across various model scales and resolutions. Notably, for ViT-Small backbone at 256 resolution under W4A4 configuration, our method achieves up to 26 FPS, showing the real-time performance.

6.5. Ablation Study

We regulate the number of calibration samples to verify the effectiveness of our proposed method, and the results are shown in Figure 8. We observe that the model performance stabilizes when the number of calibration samples reaches 32 for both W4A4 and W4A8 configuration. Additionally, activation compensation proves crucial for the W4A4.

W / A	Dataset / Method	NYUv2 [47] AbsRel ↓	δ_1 ↑	SUN RGB-D [67] AbsRel ↓	δ_1 ↑	iBims-1 [32] AbsRel ↓	δ_1 ↑	HyperSim [58] AbsRel ↓	δ_1 ↑	KITTI [19] AbsRel ↓	δ_1 ↑	vKITTI2 [19] AbsRel ↓	δ_1 ↑	DIODE Outdoor [75] AbsRel ↓	δ_1 ↑
\	Float32	0.056	0.984	0.500	0.660	0.150	0.714	0.328	0.508	0.046	0.982	0.084	0.912	0.799	0.289
W4	OBS [17]	0.059	0.981	0.508	0.626	0.151	0.718	0.327	0.501	0.049	0.980	0.091	0.893	0.793	0.287
W4A8	BrecQ [36]	0.099	0.903	0.489	0.692	0.183	0.593	0.377	0.372	0.051	0.951	0.606	0.132	0.799	0.010
	Ours	**0.058**	**0.982**	**0.394**	**0.756**	**0.157**	**0.700**	**0.322**	**0.523**	0.060	**0.965**	**0.108**	**0.846**	**0.832**	**0.298**
W4A4	BrecQ [36]	0.342	0.296	0.396	0.416	0.419	0.222	0.703	0.047	0.385	0.235	0.757	0.009	0.765	0.034
	Ours	**0.070**	**0.972**	0.466	**0.742**	**0.177**	**0.615**	**0.322**	**0.512**	**0.059**	**0.935**	**0.100**	**0.882**	**0.758**	**0.280**

Table 2. Main results of Depth Anything model with ViT-Large backbone. The first four sets are indoor scenes, while the last three are outdoor scenes. Full detailed results are included in Table A2 at Appendix 10.

W / A	Size (MB)	Res.	Latency (ms) ↓	FPS ↑	Power Eff. (GMAC/W) ↑
ViT-Small					
Float32	143.1	256	76.8 (1×)	13.02 (1×)	85.6 (1×)
W4A8	17.88		42.6 (1.8×)	23.47 (1.8×)	172.8 (2.0×)
W4A4			38.3 (2.0×)	26.11 (2.0×)	177.5 (2.1×)
Float32	143.1	512	325.5 (1×)	3.07 (1×)	87.3 (1×)
W4A8	17.88		159.0 (2.0×)	6.29 (2.0×)	200.1 (2.3×)
W4A4			133.4 (2.4×)	7.50 (2.4×)	220.3 (2.5×)
Float32	143.1	1024	2181.4 (1×)	0.46 (1×)	78.4 (1×)
W4A8	17.88		836.8 (2.6×)	1.20 (2.6×)	228.9 (2.9×)
W4A4			616.7 (3.5×)	1.62 (3.5×)	287.0 (3.7×)
ViT-Large					
Float32	1572	256	584.1 (1×)	1.71 (1×)	118.2 (1×)
W4A8	196.5		223.5 (2.6×)	4.47 (2.6×)	331.1 (2.8×)
W4A4			167.9 (3.5×)	5.96 (3.5×)	425.4 (3.6×)
Float32	1572	512	2214.5 (1×)	0.45 (1×)	126.4 (1×)
W4A8	196.5		798.4 (2.8×)	1.25 (2.8×)	392.5 (3.1×)
W4A4			568.5 (3.9×)	1.76 (3.9×)	509.3 (4.0×)
Float32	1572	1024	13297.1 (1×)	0.08 (1×)	126.9 (1×)
W4A8	196.5		4228.3 (3.1×)	0.24 (3.1×)	446.8 (3.5×)
W4A4			2725.1 (4.9×)	0.37 (4.9×)	640.6 (5.0×)

Table 3. ASIC results of Metric3D with ViT-Small and ViT-Large. ViT-Giant results are shown in Table A3 at Appendix 10.

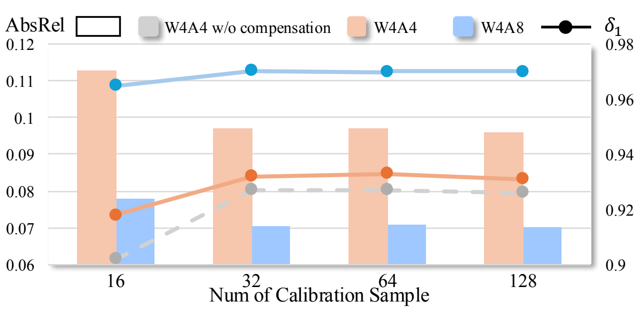

Figure 8. Accuracy ablation for the number of calibration samples using ViT-Large backbone on NYUv2 dataset.

We further perform ablation study on hardware resources by varying the number of MMUs and VCUs (i.e., number of cores). The results of 2, 4 and 8 cores are presented in Figure 9. The results show that more cores lead to faster speed. Compared to the Float32 model, the quantized model

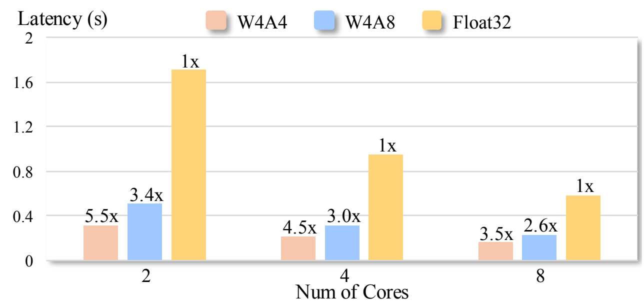

Figure 9. Latency ablation for different number of cores using ViT-Large backbone with 256x256 resolution.

is not sensitive to the hardware resource and achieves high hardware efficiency.

Meanwhile, we also visualize depth estimation results of Metric3D with ViT-Large backbone in Figure A1 at Appendix 11 compared with BrecQ [36] on NYUv2 [47] for indoor scene and KITTI [19] for outdoor scene. The visualizations verify the effectiveness of our methods on both indoor and outdoor scenes.

7. Conclusion

In this paper, we propose **QuartDepth**, a quantization framework for the real-time acceleration of MDE models on ASICs. We provide the LogNP polishing method to smooth the outliers in the decoder of MDE models, and provide the compensation method to mitigate the loss of activation quantization. We then introduce the weight reconstruction to reduce the weight quantization loss. Furthermore, we design a novel flexible and programmable hardware accelerator for our efficient deployment on ASICs. Experimental results verify the effectiveness of our proposed framework.

8. Acknowledgment

This work was mainly supported by Northeastern University and Nanjing University. This work was also supported by National Natural Science Foundation of China 62472104.

References

[1] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *arXiv e-prints*, abs/1812.11941:arXiv:1812.11941, 2018. 2

[2] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert systems with applications*, 165:113816, 2021. 2

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4009–4018, 2021. 1

[4] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023. 1, 2

[5] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[6] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*, pages 557–565. PMLR, 2017. 5

[7] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2, 2020. 7, 3

[8] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *Advances in neural information processing systems*, 29, 2016. 1

[9] Weifeng Chen, Shengyi Qian, David Fan, Noriyuki Kojima, Max Hamilton, and Jia Deng. Oasis: A large-scale dataset for single image 3d in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 679–688, 2020. 1

[10] Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017. 3

[11] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019. 2, 6, 7

[12] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, and Chen Change Loy. MeViS: A large-scale benchmark for video segmentation with motion expressions. In *ICCV*, pages 2694–2703, 2023. 2

[13] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. MOSE: A new dataset for video object segmentation in complex scenes. In *ICCV*, pages 20224–20234, 2023. 2

[14] Yifu Ding, Haotong Qin, Qinghua Yan, Zhenhua Chai, Junjie Liu, Xiaolin Wei, and Xianglong Liu. Towards accurate post-training quantization for vision transformer. In *Proceedings of the 30th ACM international conference on multimedia*, pages 5380–5388, 2022. 2

[15] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3, 7

[16] Sara Elkerdawy, Hong Zhang, and Nilanjan Ray. Lightweight monocular depth estimation model by joint end-to-end filter pruning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4290–4294. IEEE, 2019. 2

[17] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022. 6, 7, 8, 2, 3

[18] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 740–756. Springer, 2016. 2

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 6, 7, 8, 1, 2, 3

[20] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in Neural Information Processing Systems*, 31, 2018. 5

[21] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965. 4

[22] Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W Lee, et al. Aˆ 3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 328–341. IEEE, 2020. 3

[23] Tae Jun Ham, Yejin Lee, Seong Hoon Seo, Soosung Kim, Hyunji Choi, Sung Jun Jung, and Jae W. Lee. Elsa: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 692–705, 2021. 3

[24] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2018. 5

[25] Alexander Immer, Tycho van der Ouderaa, Gunnar Rätsch, Vincent Fortuin, and Mark van der Wilk. Invariance learning in deep neural networks with differentiable laplace approximations. *Advances in Neural Information Processing Systems*, 35:12449–12463, 2022. 5

[26] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, pages 2704–2713, 2018. 6, 7, 2

[27] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018. 6, 7, 2

[28] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit. *arXiv e-prints*, art. arXiv:1704.04760, 2017. 3

[29] Sheng-Chun Kao, Suvinay Subramanian, Gaurav Agrawal, Amir Yazdanbakhsh, and Tushar Krishna. Flat: An optimized dataflow for mitigating attention bottlenecks. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, page 295–310, New York, NY, USA, 2023. Association for Computing Machinery. 3

[30] Ben Keller, Rangharajan Venkatesan, Steve Dai, Stephen G. Tell, Brian Zimmer, William J. Dally, C. Thomas Gray, and Brucek Khailany. A 17–95.6 tops/w deep learning inference accelerator with per-vector scaled 4-bit quantization for transformers in 5nm. In *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pages 16–17, 2022. 3

[31] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 2

[32] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Korner. Evaluation of cnn-based single-image depth estimation methods. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 7, 8, 3

[33] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019. 5

[34] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020. 2

[35] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12, 2016. 2

[36] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. 2, 6, 7, 8, 3

[37] Yanyu Li, Pu Zhao, Geng Yuan, Xue Lin, Yanzhi Wang, and Xin Chen. Pruning-as-search: Efficient neural architecture search via channel pruning and structural reparameterization. *arXiv preprint arXiv:2206.01198*, 2022. 2

[38] Yanyu Li, Changdi Yang, Pu Zhao, Geng Yuan, Wei Niu, Jiexiong Guan, Hao Tang, Minghai Qin, Qing Jin, Bin Ren, Xue Lin, and Yanzhi Wang. Towards real-time segmentation on the edge. In *AAAI*, 2023. 2

[39] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repqvit: Scale reparameterization for post-training quantization of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17227–17236, 2023. 2

[40] Ji Lin, Wei-Ming Chen, and Song Han. *Algorithm-System Co-design for Efficient and Hardware-Aware Embedded Machine Learning*, pages 349–370. 2023. 3

[41] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1173–1179, 2022. 2

[42] Jun Liu, Zhenglun Kong, Pu Zhao, Changdi Yang, Hao Tang, Xuan Shen, Geng Yuan, Wei Niu, Wenbin Zhang, Xue Lin, Dong Huang, and Yanzhi Wang. Toward adaptive large language models structured pruning via hybrid-grained weight importance assessment. In *AAAI*, 2025. 2

[43] Liqiang Lu, Yicheng Jin, Hangrui Bi, Zizhang Luo, Peng Li, Tao Wang, and Yun Liang. Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, page 977–991, New York, NY, USA, 2021. Association for Computing Machinery. 3

[44] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015. 5

[45] John E Mottershead and Yitshak M Ram. Inverse eigenvalue problems in vibration absorption: passive modification and active control. *Mechanical systems and signal processing*, 20(1):5–44, 2006. 4

[46] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020. 2, 5, 6, 7, 1

[47] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 6, 7, 8, 1, 2, 3

[48] NVIDIA. Cuda c++ programming guidie. Website. https://docs.nvidia.com/cuda/cuda-c-programming-guide/. 6

[49] Jun-Seok Park, Changsoo Park, Suknam Kwon, Hyeong-Seok Kim, Taeho Jeon, Yesung Kang, Heonsoo Lee, Dong-woo Lee, James Kim, YoungJong Lee, Sangkyu Park, Jun-Woo Jang, SangHyuck Ha, MinSeong Kim, Jihoon Bang, Suk Hwan Lim, and Inyup Kang. A multi-mode 8k-mac hw-utilization-aware neural processing unit with a unified multi-precision datapath in 4nm flagship mobile soc. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 246–248, 2022. 3

[50] Andrea Pilzer, Stephane Lathuiliere, Nicu Sebe, and Elisa Ricci. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9768–9777, 2019. 2

[51] J.-A. Pineiro, S.F. Oberman, J.-M. Muller, and J.D. Bruguera. High-speed function approximation using a minimax quadratic interpolator. *IEEE Transactions on Computers*, 54(3):304–318, 2005. 6

[52] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019. 2

[53] Zheng Qu, Liu Liu, Fengbin Tu, Zhaodong Chen, Yufei Ding, and Yuan Xie. Dota: detect and omit weak attentions for scalable transformer acceleration. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, page 14–26, New York, NY, USA, 2022. Association for Computing Machinery. 3

[54] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 1, 2

[55] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021. 3

[56] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021. 1

[57] Alex Rasla and Michael Beyeler. The relative importance of depth cues and semantic edges for indoor mobility using simulated prosthetic vision in immersive virtual reality. In *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology*, pages 1–11, 2022. 1, 2

[58] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021. 7, 8, 3

[59] Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. Agilequant: Activation-guided quantization for faster inference of llms on the edge. In *AAAI*, 2024. 2

[60] Xuan Shen, Zhaoyang Han, Lei Lu, Zhenglun Kong, Peiyan Dong, Zhengang Li, Yanyue Xie, Chao Wu, Miriam Leeser, Pu Zhao, Xue Lin, and Yanzhi Wang. Hotaq: Hardware oriented token adaptive quantization for large language models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2024.

[61] Xuan Shen, Zhenglun Kong, Changdi Yang, Zhaoyang Han, Lei Lu, Peiyan Dong, et al. Edgeqat: Entropy and distribution guided quantization-aware training for the acceleration of lightweight llms on the edge. *arXiv preprint arXiv:2402.10787*, 2024.

[62] Xuan Shen, Pu Zhao, Yifan Gong, Zhenglun Kong, Zheng Zhan, Yushu Wu, Ming Lin, Chao Wu, Xue Lin, and Yanzhi Wang. Search for efficient large language models. In *NeurIPS*, 2024. 2

[63] Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Yanyu Li, Yifan Gong, Kai Zhang, Hao Tan, Jason Kuen, Henghui Ding, et al. Lazydit: Lazy learning for the acceleration of diffusion transformers. In *AAAI*, 2025. 2

[64] Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Jing Liu, Ruiyi Zhang, Ryan A Rossi, Hao Tan, Tong Yu, Xiang Chen, et al. Numerical pruning for efficient autoregressive models. In *AAAI*, 2025. 2

[65] Xuan Shen, Hangyu Zheng, Yifan Gong, Zhenglun Kong, Changdi Yang, Zheng Zhan, Yushu Wu, Xue Lin, Yanzhi Wang, Pu Zhao, and Wei Niu. Sparse learning for state space models on mobile. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[66] M. Song, S. Lim, and W. Kim. Monocular depth estimation using laplacian pyramid-based depth residuals. *IEEE Transactions on Circuits and Systems for Video Technology*, 31 (11):4381–4393, 2021. 1, 2

[67] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015. 7, 8, 3

[68] Harshit Suri, Harshit Mahajan, Kartik Kumar Chauhan, Aman Anand, and Subrata Sahana. *Computer Vision: A Detailed Review on Augmented Reality (AR), Virtual Reality (VR), Telehealth, and Digital Radiology*, pages 99–115. Springer Nature Singapore, Singapore, 2023. 2

[69] Chellammal Surianarayanan, John Jeyasekaran Lawrence, Pethuru Raj Chelliah, Edmond C. Prakash, and Chamin Nalinda Lokugam Hewage. A survey on optimization techniques for edge artificial intelligence (ai). *Sensors (Basel, Switzerland)*, 23, 2023. 2

[70] Synopsys. Design compiler concurrent timing, area, power, and test optimization. Website, . https://www.synopsys.com/implementation-and-

signoff/rtl-synthesis-test/dc-ultra.html. 7
[71] Synopsys. Primetime static timing analysis. Website, . https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html. 7
[72] Fengbin Tu, Shouyi Yin, Peng Ouyang, Shibin Tang, Leibo Liu, and Shaojun Wei. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(8):2220–2233, 2017. 3
[73] Shikhar Tuli and Niraj K. Jha. Acceltran: A sparsity-aware accelerator for dynamic inference with transformers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(11):4038–4051, 2023. 3
[74] Kodai Ueyoshi, Ioannis A. Papistas, Pouya Houshmand, Giuseppe M. Sarda, Vikram Jain, Man Shi, Qilin Zheng, Sebastian Giraldo, Peter Vrancx, Jonas Doevenspeck, Debjyoti Bhattacharjee, Stefan Cosemans, Arindam Mallik, Peter Debacker, Diederik Verkest, and Marian Verhelst. Diana: An end-to-end energy-efficient digital and analog hybrid neural network soc. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 1–3, 2022. 3
[75] Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. DIODE: A Dense Indoor and Outdoor DEpth Dataset. *CoRR*, abs/1908.00463, 2019. 7, 8, 3
[76] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International conference on machine learning*, pages 6566–6575. PMLR, 2019. 5
[77] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019. 1, 2
[78] Yang Wang, Yubin Qin, Dazheng Deng, Jingchuan Wei, Yang Zhou, Yuanqi Fan, Tianbao Chen, Hao Sun, Leibo Liu, Shaojun Wei, and Shouyi Yin. A 28nm 27.5tops/w approximate-computing-based transformer processor with asymptotic sparsity speculating and out-of-order computing. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 1–3, 2022. 3
[79] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*, 2022. 2
[80] Li-Hua Wen and Kang-Hyun Jo. Deep learning-based perception systems for autonomous driving: A comprehensive survey. *Neurocomputing*, 489:255–270, 2022. 2
[81] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108. IEEE, 2019. 2
[82] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 1, 2
[83] Chao Wu, Yifan Gong, Liangkai Liu, Mengquan Li, Yushu Wu, Xuan Shen, Zhimin Li, Geng Yuan, Weisong Shi, and Yanzhi Wang. Aye-edge: Automated deployment space search empowering accuracy yet efficient real-time object detection on the edge. *arXiv preprint arXiv:2408.05363*, 2024. 2
[84] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020. 2
[85] Yushu Wu, Yifan Gong, Pu Zhao, et al. Compiler-aware neural architecture search for on-mobile real-time super-resolution. In *ECCV*, pages 92–111. Springer, 2022. 2
[86] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular relative depth perception with web stereo data supervision. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 311–320, 2018. 1
[87] Ke Xian, Jianming Zhang, Oliver Wang, Long Mai, Zhe Lin, and Zhiguo Cao. Structure-guided ranking loss for single image depth prediction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1
[88] Changdi Yang, Pu Zhao, Yanyu Li, et al. Pruning parameterization with bi-level optimization for efficient semantic segmentation on the edge. In *CVPR*, pages 15402–15412, 2023. 2
[89] Guanglei Yang, Hao Tang, Mingli Ding, Nicu Sebe, and Elisa Ricci. Transformer-based attention networks for continuous pixel-wise prediction. In *Proceedings of the IEEE/CVF International Conference on Computer vision*, pages 16269–16279, 2021. 1
[90] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 1, 2, 7, 3
[91] Wei Yin, Yifan Liu, and Chunhua Shen. Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7282–7295, 2021. 1
[92] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (CVPR)*, 2021. 1, 2
[93] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 204–213, 2021. 1
[94] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Simon Chen, Yifan Liu, and Chunhua Shen. Towards accurate reconstruction of 3d scene shape from a single monocular image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6480–6494, 2022. 1
[95] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d:

Towards zero-shot metric 3d prediction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9043–9053, 2023. 1, 2, 3, 7

[96] Haoran You, Zhanyi Sun, Huihong Shi, Zhongzhi Yu, Yang Zhao, Yongan Zhang, Chaojian Li, Baopu Li, and Yingyan Lin. Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 273–286, 2023. 2, 3

[97] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *ICLR*, 2020. 1, 2

[98] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020. 2

[99] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3916–3925, 2022. 1

[100] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization framework for vision transformers with twin uniform quantization. *arXiv preprint arXiv:2111.12293*, 2021. 2

[101] Zheng Zhan, Yifan Gong, Pu Zhao, Geng Yuan, et al. Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In *ICCV*, pages 4821–4831, 2021. 2

[102] Zheng Zhan, Zhenglun Kong, Yifan Gong, Yushu Wu, Zichong Meng, Hangyu Zheng, Xuan Shen, Stratis Ioannidis, Wei Niu, Pu Zhao, and Yanzhi Wang. Exploring token pruning in vision state space models. In *NeurIPS*, 2024. 2

[103] Zheng Zhan, Yushu Wu, Yifan Gong, Zichong Meng, Zhenglun Kong, Changdi Yang, Geng Yuan, Pu Zhao, Wei Niu, and Yanzhi Wang. Fast and memory-efficient video diffusion using streamlined inference. In *NeurIPS*, 2024. 2

[104] Zheng Zhan, Yushu Wu, Zhenglun Kong, Changdi Yang, Yifan Gong, Xuan Shen, Xue Lin, Pu Zhao, and Yanzhi Wang. Rethinking token reduction for state space models. In *EMNLP*, pages 1686–1697, Miami, Florida, USA, 2024. ACL. 2

[105] Chi Zhang, Wei Yin, Billzb Wang, Gang Yu, Bin Fu, and Chunhua Shen. Hierarchical normalization for robust monocular depth estimation. *Advances in Neural Information Processing Systems*, 35:14128–14139, 2022. 1, 2

[106] Yihua Zhang, Yuguang Yao, Parikshit Ram, Pu Zhao, Tianlong Chen, Mingyi Hong, Yanzhi Wang, and Sijia Liu. Advancing model pruning via bi-level optimization. *Advances in Neural Information Processing Systems*, 35:18309–18326, 2022. 2

[107] Pu Zhao, Xuan Shen, Zhenglun Kong, Yixin Shen, Sung-En Chang, Timothy Rupprecht, Lei Lu, Enfu Nan, Changdi Yang, Yumei He, et al. Fully open source moxin-7b technical report. *arXiv preprint arXiv:2412.06845*, 2024. 1

[108] Pu Zhao, Fei Sun, Xuan Shen, Pinrui Yu, Zhenglun Kong, Yanzhi Wang, and Xue Lin. Pruning foundation models for high accuracy without retraining. In *Findings of EMNLP 2024*, pages 9681–9694. ACL, 2024. 2