SpRDA: An Efficient Sparse Accelerator for Robotics Diffusion Model

Renda Jian, Yuzhou Chen, Pan Zhao, Yitian Chen, Zhican Wang, Xueling Wang, Chen Zhang, Guanghui He*
School of Integrated Circuits, Shanghai Jiao Tong University
Shanghai, China
{Jian_rd,huygens,panzhao08,kaorimsly,wang_zhican,wangxueling02,chenzhang.sjtu,guanghui.he}@sjtu.edu.cn

Abstract

Diffusion models have recently demonstrated great performance in robotic manipulation, standing out with their ability to handling multi-modal action distributions and high-dimensional action spaces. However, its iterative computing feature limits real-time control in resource-constrained robot platform. Some of existing diffusion accelerators explored sparsity based on the similarity of adjacent steps, but suffering from low sparsity of full network and hardware inefficiency due to under-utilization of processing element (PE) and ignorance of more significant non-matrix-multiplication operators (NMOs) in sparse model. To address these issues, we propose SpRDA, an algorithm-architecture co-design accelerator that can achieve high-speed and energy-efficient inference of robotics diffusion model. A fine-grained and difference-aware cache algorithm combining differential computing and cache is proposed to identify more redundant computation in robotics diffusion, achieving over 90% sparsity with no distinct accuracy degradation. Furthermore, we apply two-level group sparsity adjusting and design a dynamic sparsity-aware matrix processing array to fully leverage sparsity in hardware. We also propose a graph-based vector unit to process various NMOs in higher throughput and lower area. Compared to the state-of-the-art diffusion accelerators, SpRDA achieves 1.56-2.60× speedup and 6.45-7.85× energy efficiency improvement.

Keywords

Embodied Artificial Intelligence, Diffusion Model, Sparsity Exploitation, Algorithm-Architecture Co-Design

1 Introduction

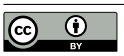
Embodied Artificial Intelligence (Embodied AI) is pushing traditional robots over their past boundary, autonomously tackling an ever-widening range of tasks [14]. Different from large language models, Embodied AI generate actions to control robots to interact with the environment. Diffusion models have been widely applied in robotic trajectory generation for its superior performance over other generative models and traditional methods [19, 22]. Diffusion-based imitation learning have demonstrated its ability to model complex multi-modal action distributions and excellent scalability to high-dimensional output spaces [3, 13]. Diffusion Transformer (DiT) [16] architecture particularly excels at handling complex tasks [2].

Despite their advanced capacity, diffusion models suffer from higher computational costs and inference latency. The inference starts at random noise and the overall denoising process requires several sequential steps. The inference of Diffusion Policy (DP) involves up to 100 steps [3] and each step involves a significant number of operations. When applying in the edge resource-constrained robot platform, this issue becomes more severe causing difficulty in achieving real-time and high-frequency control. Recent robotic diffusion models have trended toward fewer sampling steps, yet their latency remains beyond the tolerable range for practical robotic applications. On GeForce GPUs, DP with 10 steps achieves a control frequency of only 10Hz [3], while RDT with 5 steps reaches 6Hz [13].

Existing acceleration for diffusion models mainly focus on image and video generation. Prior work has demonstrated high similarity among the outputs of adjacent iterations, forming the foundation for most training-free methods. Cache is a simple acceleration on GPU, reducing computation by directly reusing the results of attention module and feed forward network (FFN) every N steps [17, 23]. However, the parameter N is difficult to tune and our experiments reveal that applying cache to DP leads to a reduction in the average success rate of up to 68%. Specialized accelerators for diffusion models mostly apply low bit-width or sparsity to reduce redundant computation. Differential computing constrains input differences to a narrower range, enabling low-bit computation [10]; however, this is accompanied by 2× increase in memory access overhead. Specifically, differential computing can only quantize 50–90% of these differences into half-precision format, and even such half-precision arithmetic can reduce computational complexity by at most half.

Sparse computing serves as a more efficient approach for diffusion model acceleration, yet it is plagued by critical issues. **First, existing diffusion sparsity generation algorithms fail to fully leverage the similar characteristics, resulting in lower overall network sparsity.** Ditto [9] produces only 5-30% sparsity by differential computing in each step. EXION [7] leverages output sparsity; however, within the attention block, it fails to exploit diffusion-specific features and instead relies on traditional attention prediction. The error accumulation across steps constrains the achievement of higher sparsity. Moreover, the input and output projection operations that account for a substantial proportion of

*Corresponding author.



DAC '26, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3804269

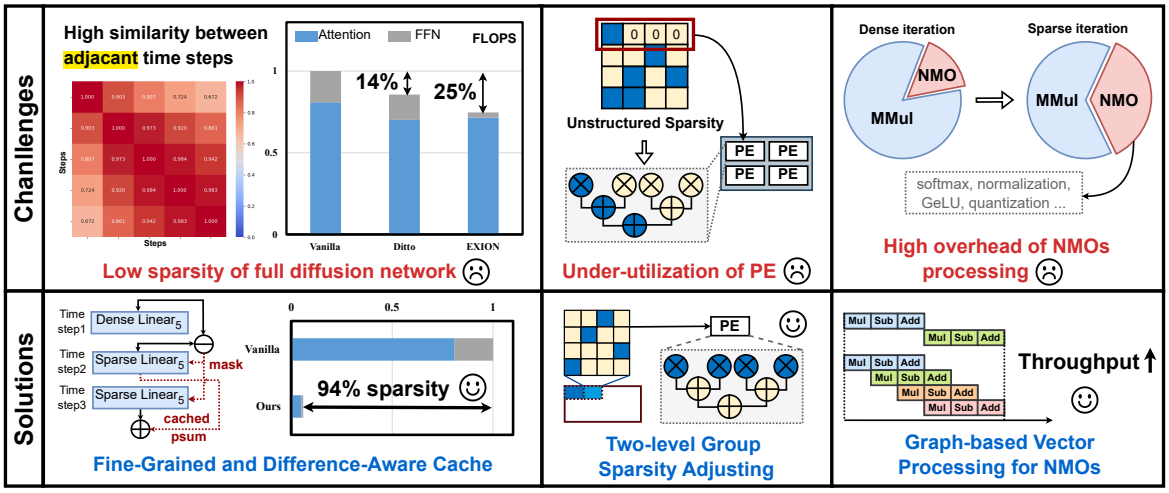


Figure 1: Challenges and solutions of sparsity in robotics diffusion model.

computation in robotics diffusion models, derive minimal benefit from EXION's sparsity mechanism.

Second, high unstructured sparsity fails to translate into substantial hardware speedup, primarily due to the low utilization of processing elements (PEs). The fixed sparse mode of GPU is not flexible enough for algorithm. The sparsity in Ditto is generated following element-wise subtraction, which not only tends to cause workload imbalance among PEs but also limits the potential for further zero-skipping. EXION propose a matrix compression method yet it still achieves only 50% hardware utilization in some cases while incurring high overhead in merging and dataflow control. Additionally, it cannot handle input sparsity.

Third, there are a variety of non-matrix-multiplication operators (NMOs) in diffusion models, such as softmax, normalization, non-linear activation function and dynamic quantization, with high overhead of floating-point arithmetic. **As matrix multiplication grow sparser, NMOs account for an increasing proportion of overall latency, a factor always overlooked by prior sparse accelerators.**

To solve these challenges, we propose an algorithm-architecture co-design, SpRDA. The key contributions are as follows:

- We propose *a fine-grained and difference-aware cache algorithm*, combining differential computing and direct cache, applied in all linear layers. Based on the difference of the first two steps, it identifies the unimportant input positions during denoising to get the sparse mask, while subsequent steps reuse the mask and partial sums of unimportant inputs. Our approach can achieve over 90% sparsity with less than 1% performance degradation.
- We introduce *two-level group sparsity adjusting* to bridge sparse algorithm and hardware. This mechanism not only ensures flexible sparsity variations within a matrix, but also enables each tile to maintain structured sparsity. Additionally, we design a dynamic sparsity-aware matrix processing array to fully eliminate computational redundancy.
- We conduct a detailed analysis of the computation of all involved NMOs and design *a graph-based vector processing array* comprising a set of basic arithmetic units with configurable interconnections. It can be configured to handle the dataflow of various NMOs with high throughput.

The SpRDA is implemented on a 28nm technology and fully evaluated with widely-used robotics diffusion models [3, 13] on a set of tasks. Compared with the prior state-of-the-art accelerators [7, 9], our work achieves 1.56-2.60× speedup and 6.45-7.85× improvement on energy efficiency.

2 Background and Motivation

2.1 Robotics Diffusion Model

Diffusion models, based on the concept of Markov chains, transform the complex problem of generating target distributions into a step-by-step denoising process [8]. The reverse denoising process starts from a simple noise distribution. Over a series of time steps, the noise is iteratively reduced, causing the distribution to gradually approach the true distribution. At each time step, the noise prediction model uses a neural network to predict the direction and extent of denoising based on the output of last step.

Robotics diffusion models represent a robot's action policy as a conditional denoising process [3], as Figure 2 shows. The trajectory of the robot arm gradually transitions from a random distribution in the action space to the correct path toward the target. The action sequence serves as both the output of the model and the input for the next denoising step. The backbone of model is mainly constructed by several stacked DiT blocks, each consisting of three major modules: self-attention, cross-attention and FFN. Figure 2 exhibits the operations in cross-attention and FFN modules. The cross-attention modules take observed images or language instructions as conditions. Once an inference is completed, the robot arm performs actions, and the camera captures a new image to initiate the next inference cycle, continuing until the task is successful or the maximum number of action steps is reached.

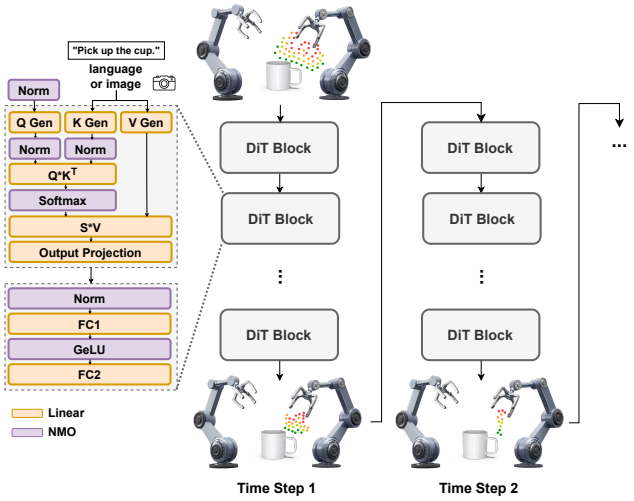


Figure 2: The architecture of robotics diffusion model. We show the operations in cross-attention and FFN modules; self-attention is similar to cross-attention in operation structure.

2.2 Motivation

Unlike image and video generation models, robotics diffusion models involve fewer tokens. This is because each token represents only one action and each inference cannot generate an excessive number of actions that ensures adaptability to potential environmental changes. Thus the linear computation with weight accounts for over 99% matrix multiplication with only 1% attention computation. Though the image and language conditions may involve more tokens, the value of K (keys) and V (values) remain constant across denoising steps, allowing K and V to be treated as weights.

In the action denoising process, outputs of adjacent steps hold a high similarity as Figure 3(a) exhibit. That means each step in fact contributes a small part of change, and there exists much redundant operations. Further analysis reveals that the similarity is evident not only in the values but also in the distribution of differences. Figure 3(b) shows that the input differences between each step and first step in RDT. The differences gradually increase with steps but the distribution pattern is similar. The positions with larger difference in second step tend to continue increasing the difference in next steps.

3 Sparsification Strategy for Robotics Diffusion Models

In this section, we propose an adaptive and effective sparse algorithm based on iteration similarity to accelerate robotics diffusion models. It combines Fine-grained and Difference-Aware Cache (FDC) with Two-level Group Sparsity Adjusting (TGSA). FDC identifies unimportant input positions according to the difference of first two steps, reusing unimportant partial sums to reduce redundancy and achieve high sparsity. TSA transforms unstructured sparsity to a hardware-efficient sparse pattern.

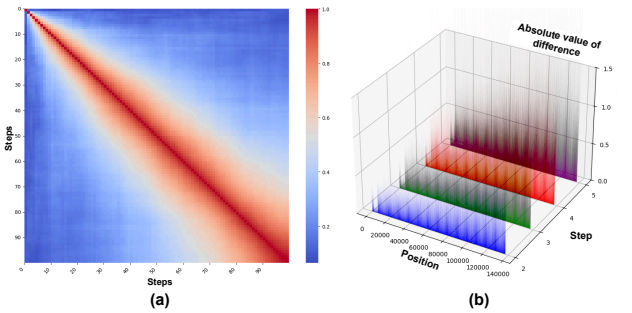


Figure 3: (a) The output similarity heatmap of DP layer1 in 100 steps. (b) The FFN input differences with first step in RDT.

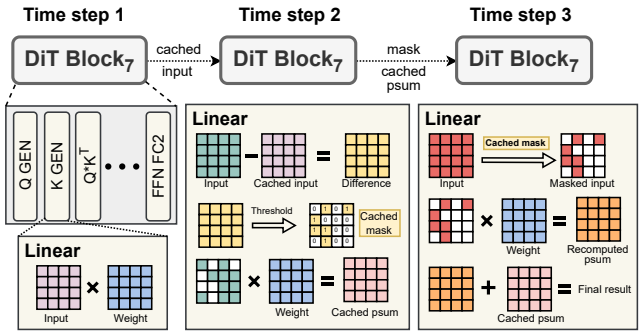


Figure 4: Detailed workflow of fine-grained and difference-aware cache.

3.1 Fine-Grained and Difference-Aware Cache

To mitigate numerous operations in diffusion model, we need to first determine the excessive computation based on the similarity analyzed in Sec 2.2. The prior coarse-grained cache method [17, 23] simply treats the whole attention/FFN modules as redundancy, causing significant performance degradation. Therefore a more fine-grained identification is required.

Focusing on the bottleneck linear layers, inputs with minimal differences from the first step can be deemed unnecessary. This is because data with negligible difference will converge to the same value after quantization, while weights remain unchanged throughout denoising. Moreover, we have known the difference distribution is also similar across denoising steps so the important positions can be confirmed in second step. We propose fine-grained and difference-aware cache acceleration as depicted in Figure 4.

FDC is applied in all linear layers including attention computation of cross-attention. In the second step, it calculates the difference between first two steps and generates the sparse mask based on threshold: 1 denotes critical positions with large difference, while 0 indicates unimportant positions to produce cached partial sum (psum). In subsequent steps, the generated mask is reused to sparsify the linear layers, only activating computations for critical positions. The final result is obtained by accumulating current sparse computation outputs with the cached psums.

Algorithm 1 Two-level Group Sparsity Adjusting

Input: Difference matrix $D \in \mathbb{R}^{T \times C}$, group size $T_G, C_G \in \mathbb{N}$, tile size $W_{\text{tile}} \in \mathbb{N}$, allowed sparsity S,threshold $T \in \mathbb{R}$
Output: Sparse mask M

$N_G = \lceil T/T_G \rceil * \lceil C/(C_G * W_{\text{tile}}) \rceil$
Divide D into N_G groups including G_i (for $i \in N_G$)
for $i \in N_G$ **do**
  $M_i = (G_i > T)$
  $S_{\text{mean}} = Mean(M_i)$
  Set S_{mean} as the closest value in S
  for $tile_j$ in G_i **do**
    Pick indexes of top $(S_{\text{mean}} * W_{\text{tile}})$ values as M_{ij}
  end for
end for
$M = concat(M_{ij})$ for each tile

3.2 Two-level Group Sparsity Adjusting

To fully leverage sparsity generated in FDC, we introduce two-level sparsity [12] to sparse robotics diffusion model. In sparse matrix, different regions exhibit varying sparsity, with zeros tending cluster in specific channels or tokens. TGSA partitions sparse matrices into multiple groups with varying sparsity, where each tile within a group exhibits a consistent pattern. The first-level flexible groups align with the sparse feature of algorithm, while second-level tiles with fixed pattern facilitate high hardware efficiency.

TGSA only changes the method of generating mask and almost has no impact on sparsity and model performance. Its detailed adjusting method is depicted in Algorithm 1. After dividing into groups, it first gets the raw mask according to threshold and finds the closest value of mean sparsity in allowed sparsity. In practice, we set W_{tile} as 8 and S as 0,2,4,8, corresponding to sparsity of 100%, 75%, 50%, 0%. Finally, TGSA picks up indexes of the largest values in each tile as recomputed positions.

4 Efficient Sparse Accelerator

In this section, we present the design of an efficient sparse accelerator for robotics diffusion model to maximize the hardware speedup brought by software sparsity.

4.1 Overview

Figure 5 depicts the overall architecture of SpRDA. It primarily consists of 112KB on-chip buffer, a dynamic sparsity-aware matrix processing array (DSMA), and a graph-based vector processing array (GVA). The on-chip buffer is divided into five main parts for input feature maps, weights, temporary results, sparse masks and reuse vectors respectively. Reuse vectors refer to input of first step or cached psum of second step. The SMA is composed of 64 matrix processing units (MPU), each consisting 4 matrix processing elements (MPE). The GVA is mainly responsible for processing NMOs.

4.2 Dynamic Sparsity-Aware Matrix Processing Array

The mask of a group generated by TGSA contains a sparse pattern (encoded in 2 bits), index of non-zero value and position of the

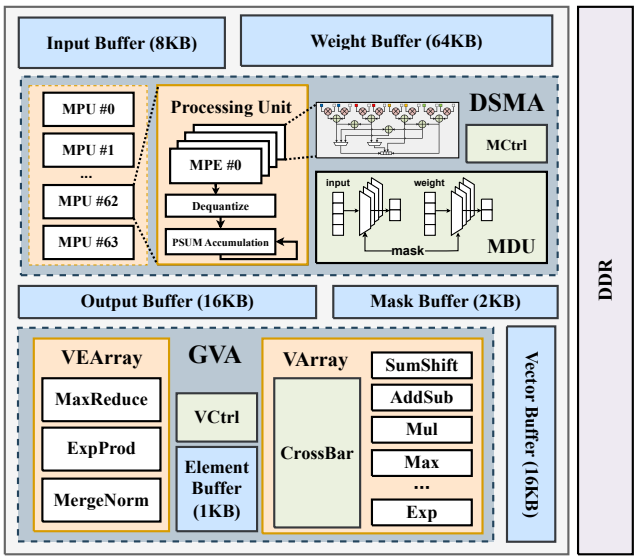


Figure 5: Overview of SpRDA architecture.

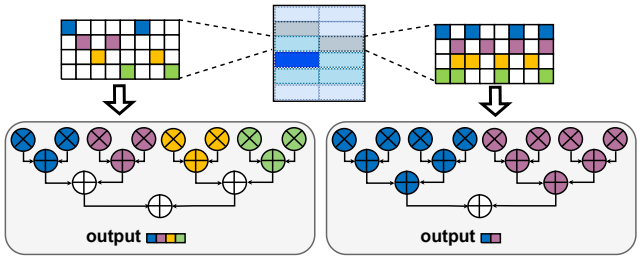


Figure 6: The sparsity-aware PE design.

group. This enable the controller to read only data from groups with non-100% sparsity from buffer, allowing the array fully skip the unnecessary overhead of zero values. The mask decoding unit (MDU) is responsible for selecting corresponding inputs and weights.

As Figure 6 shown, we propose a configurable PE design to support both two-level sparse and dense matrix multiplication. The PE consists of 8 multipliers and an 8-to-1 adder tree. Compared to traditional multiply-and-accumulate (MAC) operation, this design adds bypass paths between each layer of the adder tree and output. For example, the sparse pattern depicted in the left part of Figure 6 is 2:8 which means there are two non-zero values of eight inputs in a row. PE allocates 2 multipliers per row and the outputs are generated at the first layer of adder tree via the forwarding path, therefore PE computes results for four rows in a single cycle. If the pattern is 4:8 as shown in the right part of Figure 6, PE generates two results from the second layer in a cycle. When sparsity is 0%, the dataflow of PE is the same as traditional.

4.3 Graph-Based Vector Processing Array

In robotics diffusion model, apart from matrix multiplication (MMul), there are various NMOs including softmax in attention, non-linear

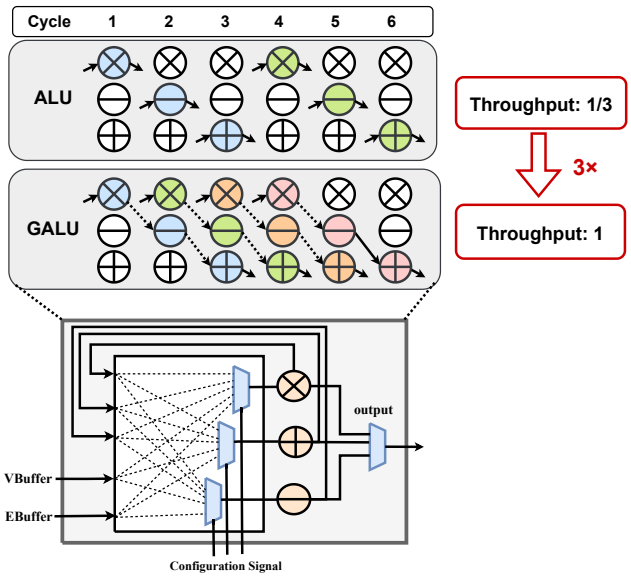


Figure 7: A simplified version of VArray and the dataflow comparison of ALU and GALU.

activation function in FFN, normalization, residual adding and some operations for MMul acceleration like dynamic quantization and mask generation mentioned in Section 3.2. Unlike MMul, NMOs employ floating-point arithmetic with large hardware overhead and long latency. Especially for sparse model, as computation of MMul is mitigated, NMOs are gradually becoming a more significant part of overall latency.

We conduct a deep analysis of the detailed computation process of all involved NMOs, decomposing them into several basic vector floating-point operations, such as vector division, vector summation and so on. Based on the idea proposed in [20], these basic units can be shared when processing different NMOs. We propose graph-based vector processing (GVP) to organize function units into a dataflow graph with a configurable interconnection. The simplified version is shown in Figure 7.

As Figure 7 depicted, leveraging interconnection between function units, gragh-based ALU (GALU) acheives higher throughput than traditional ALU. Compared to special function unit designed for certain operators, GVA excels in higher flexibilty and lower area due to time-division multiplexing.

The original softmax and layer normalization (LayerNorm) require waiting for the entire row of data before computing can start. To mitigate this latency gap, we apply a two-stage processing method [21]: the first stage computes the maximum value for softmax or the mean and variance for LayerNorm, while the second stage is responsible for normalization. In the first stage, vectors are converted to local values in vector array (VArray). Local values are then fed into the element processing unit (VEArray) to derive global value by element-dimension computing. Moreover, if an entire row of sparse matrix is empty, SpRDA skips psum computation and directly reuses the cached result after softmax or LayerNorm, yielding a further improvement in speedup.

Table 1: Comparison with cache acceleration

Model	Task	Vanilla	Cache		Ours	
		score	score	FLOPS↓	score	FLOPS↓
DP	pushT	0.94	0.85	75%	0.93	76.3%
	lift(mh*)	1.00	0.18	75%	1.00	74.1%
	lift(ph)	1.00	0.42	75%	1.00	74.7%
	can(mh)	1.00	0.08	66%	0.98	70.7%
	can(ph)	0.98	0.01	66%	0.97	72.4%
	mean	**0.984**	**0.308**	**71.4%**	**0.976**	**73.6%**
RDT	PickCube	0.792	0.724	60%	0.780	56.0%
	PegInsertion	0.100	0.068	60%	0.092	54.7%
	StackCube	0.788	0.780	60%	0.788	55.7%
	PlugCharger	0.016	0.008	60%	0.012	57.2%
	mean	**0.424**	**0.395**	**60%**	**0.418**	**55.9%**

* mh: multi-human demonstration dataset, ph: proficient-human demonstration dataset.

5 Evaluation

5.1 Evaluation Methodology

We evaluate SpRDA on two widely-used robotics diffusion models: Transformer-based Diffusion Policy (DP) [3] and RDT [13]. DP is evaluated on five tasks from two benchmarks for average score of 100 different environment initial conditions. PushT is adapted from IBC [4] and four remaining tasks are from Robomimic benchmark [15]. RDT is evaluated on four tasks from ManiSkill simulation benchmark [5] for average success rate of 10 random seeds with 25 trials per seed.

For hardware implementation, we implement the compute and control logic of SpRDA with SystemVerilog and estimate the area and energy consumption of on-chip memory with CACTI [1]. The prototype is synthesized with Synopsys Design Compiler under TSMC 28nm CMOS technology. The performance of sparse computing is based on the dynamic sparsity acquired from DP (pushT task) and RDT.

Utilizing post-training quantization, all baseline and SpRDA hardware are designed to execute 8-bit activation and 8-bit weight (A8W8) quantized models, as this configuration preserves accuracy in diffusion models [6, 11, 18].

5.2 Evaluation on Proposed Algorithm

We compare our method combining FDC and TGSA with traditional cache algorithm [17, 23] in Table 1. We measure the FLOPS decrease of our method with less than 1% performance degradation and evaluate performance of cache acceleration with similar FLOPS reduction. Our algorithm achieves 73.6% computation decrease in DP, and 55.9% computation decrease in RDT. Compared to cache, we increase average score by 66.8% in DP and 2.3% in RDT.

5.3 Evaluation on Proposed Accelerator

Figure 8 presents latency evaluations over three key techniques: FDC, TGSA and GVP. Compared to the baseline, FDC introduces high sparsity to all linear layers, yielding over 32.5% reduction in

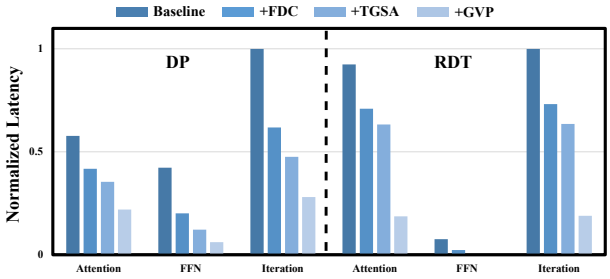


Figure 8: Normalized latency of submodules and iteration in DP and RDT through FDC, TGSA and GVP. Note that "+" means this method is appended on the previous approach. Baseline has the same hardware architecture with these functions deactivated.

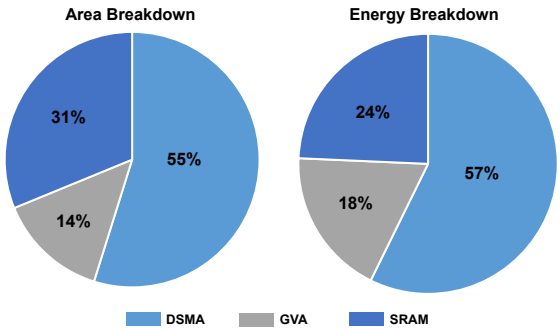


Figure 9: Area breakdown and energy breakdown of SpRDA.

latency of a sparse iteration. TGSA improves hardware utilization, leading to a 12% latency decrease. Notably, FDC and TGSA achieve lower speedup in the attention block than FFN. This is because the attention block involves more NMOs, which limits the potential for overall speedup. In contrast, the introduced GVP enhances throughput of NMOs and further decreases latency by 31.5%.

5.4 Comparison with SOTA Accelerators

We conduct a comparison of SpRDA with leading diffusion accelerators exploiting sparsity[7, 9], and ensure same configuration of PE and vector processing unit. Compared with them, SpRDA generate higher sparsity in full network by FDC and effectively improves PE utilization by employing TGSA. Moreover, utilizing GVA, SpRDA significantly increases throughput of NMOs. As a result, SpRDA improves throughput by 1.56-2.60× in Figure 10.

Table 2 presents the detailed implementation results. Utilizing TSMC 28nm technology, the SpRDA prototype demonstrates an effective performance of 5042.9 GOPS/W at 500 MHz, with a power consumption of 398 mW. The accelerator occupies an area of 1.86 mm^2. We also reconstruct the method of [7, 9] for robotics diffusion model inference to evaluate their energy efficiency. Leveraging FDC, TGSA and GVA, SpRDA shows a 6.45-7.85× increase in energy efficiency compared to these accelerators.

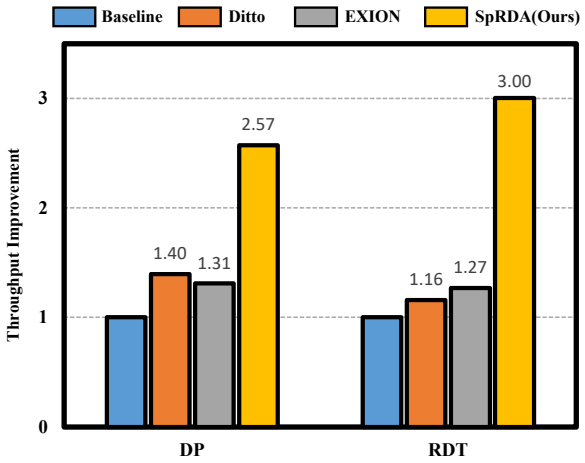


Figure 10: Normalized throughput improvements of SpRDA with SOTA diffusion accelerators [7, 9]. We ensure the peak throughput is consistent across all works to make a fair comparison.

Table 2: Comparison with other diffusion accelerators

	Ditto[9]	EXION[7]	**SpRDA**
Technology(nm)	45	14	28
Frequency(MHz)	1000	800	500
Area(mm^2)	64.48	4.37	1.86
SRAM(MB)	192	0.88	0.11
Throughput(GOPS)*	10172.2	3882.6	2007.3
Power(W)	33.6	1.5	0.4
Energy Efficiency† (GOPS/W)	781.96	642.2	**5042.9**

* Evaluation on RDT considering sparsity.
† Technology normalized to 28nm as follows: s = Technology/28nm, $f \sim s$, and $P \sim (1/s)$, where f and P is denoted as frequency and power, respectively.

6 Conclusion

This paper introduces an algorithm-architecture co-optimization, SpRDA. The key insight is to exploit higher sparsity of full network and efficiently process the sparse computing for robotics diffusion model. Through fine-grained and difference-aware cache, SpRDA achieves over 90% sparsity and applies two-level group sparsity adjusting to maximize PE utilization. Besides, SpRDA hardware accelerator integrates graph-based vector processing unit to improve throughput of NMOs. Comprehensive evaluations demonstrate SpRDA surpasses SOTA works in various metrics such as accuracy, throughput, and energy efficiency across diverse tasks and model types.

Acknowledgments

This work was supported by the New Generation Artificial Intelligence-National Science and Technology Major Project under Grant 2025ZD0122400 and National Natural Science Foundation of China under Grant U25B2057.

References

[1] Rajeev Balasubramonian, Andrew B. Kahng, Naveen Muralimanohar, Ali Shafiee, and Vaishnav Srinivas. 2017. CACTI 7: New Tools for Interconnect Exploration in Innovative Off-Chip Memories. *ACM Trans. Archit. Code Optim.* 14, 2, Article 14 (June 2017), 25 pages. doi:10.1145/3085572

[2] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. 2025. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734* (2025).

[3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2025. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research* 44, 10-11 (2025), 1684–1704.

[4] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. 2022. Implicit behavioral cloning. In *Conference on robot learning*. PMLR, 158–168.

[5] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. 2023. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. In *International Conference on Learning Representations*.

[6] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. 2023. PTQD: accurate post-training quantization for diffusion models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 580, 13 pages.

[7] Jaehoon Heo, Adiwena Putra, Jieon Yoon, Sungwoong Yune, Hangyeol Lee, Ji-Hoon Kim, and Joo-Young Kim. 2025. EXION: Exploiting Inter-and Intra-Iteration Output Sparsity for Diffusion Models. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 324–337. doi:10.1109/HPCA61900.2025.00034

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 574, 12 pages.

[9] Sungbin Kim, Hyunwuk Lee, Wonho Cho, Mincheol Park, and Won Woo Ro. 2025. Ditto: Accelerating Diffusion Model via Temporal Value Similarity. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 338–352. doi:10.1109/HPCA61900.2025.00035

[10] Weihao Kong, Yifan Hao, Qi Guo, Yongwei Zhao, Xinkai Song, Xiaqing Li, Mo Zou, Zidong Du, Rui Zhang, Chang Liu, Yuanbo Wen, Pengwei Jin, Xing Hu, Wei Li, Zhiwei Xu, and Tianshi Chen. 2025. Cambricon-D: Full-Network Differential Acceleration for Diffusion Models. In *Proceedings of the 51st Annual International Symposium on Computer Architecture* (Buenos Aires, Argentina) *(ISCA '24)*. IEEE Press, 903–914. doi:10.1109/ISCA59077.2024.00070

[11] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. 2023. Q-Diffusion: Quantizing Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 17535–17545.

[12] Jun Liu, Guohao Dai, Hao Xia, Lidong Guo, Xiangsheng Shi, Jiaming Xu, Huazhong Yang, and Yu Wang. 2023. TSTC: Two-Level Sparsity Tensor Core Enabling both Algorithm Flexibility and Hardware Efficiency. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9. doi:10.1109/ICCAD57390.2023.10323775

[13] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. 2024. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864* (2024).

[14] Yang Liu, Weixing Chen, Yongjie Bai, Xiaodan Liang, Guanbin Li, Wen Gao, and Liang Lin. 2025. Aligning Cyber Space With Physical World: A Comprehensive Survey on Embodied AI. *IEEE/ASME Transactions on Mechatronics* (2025), 1–22. doi:10.1109/TMECH.2025.3574943

[15] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. 2022. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *Proceedings of the 5th Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 164)*, Aleksandra Faust, David Hsu, and Gerhard Neumann (Eds.). PMLR, 1678–1690. https://proceedings.mlr.press/v164/mandlekar22a.html

[16] William Peebles and Saining Xie. 2023. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 4195–4205.

[17] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. 2024. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425* (2024).

[18] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. 2023. Post-Training Quantization on Diffusion Models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1972–1981. doi:10.1109/CVPR52729.2023.00196

[19] Mingchen Song, Xiang Deng, Zhiling Zhou, Jie Wei, Weili Guan, and Liqiang Nie. 2025. A survey on diffusion policy for robotic manipulation: Taxonomy, analysis, and future directions. *Authorea Preprints* (2025).

[20] Wenqiang Wang, Yuzhou Chen, Guanting Huo, Guanghui He, and Ningyi Xu. 2024. VEGA: Implementing a Versatile and Efficient Deep Learning Processor with Graph-Based ALU. In *2024 IEEE 42nd International Conference on Computer Design (ICCD)*. 591–598. doi:10.1109/ICCD63220.2024.00095

[21] Zhican Wang, Hongxiang Fan, and Guanghui He. 2025. DESA: Dataflow Efficient Systolic Array for Acceleration of Transformers. *IEEE Trans. Comput.* 74, 6 (2025), 2058–2072. doi:10.1109/TC.2025.3549621

[22] Rosa Wolf, Yitian Shi, Sheng Liu, and Rania Rayyes. 2025. Diffusion models for robotic manipulation: a survey. *Frontiers in Robotics and AI* 12 (09 2025). doi:10.3389/frobt.2025.1606247

[23] Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. 2025. EfficientVLA: Training-Free Acceleration and Compression for Vision-Language-Action Models. *arXiv preprint arXiv:2506.10100* (2025).