DWIP: Efficient Depth–Width Integrated Pruning for Vision-Language-Action Models in Robotic Manipulation

1st Qian Zhang*
National Key Laboratory of Parallel and Distributed Computing National University of Defense Technology
Changsha, China
zhangqiantlom@nudt.edu.cn

1st Rongchun Li*
National Key Laboratory of Parallel and Distributed Computing National University of Defense Technology
Changsha, China
rongchunli@nudt.edu.cn

2nd Peng Qiao†
National Key Laboratory of Parallel and Distributed Computing National University of Defense Technology
Changsha, China
pengqiao@nudt.edu.cn

***Abstract*—Vision-Language-Action (VLA) models demonstrate excellent performance in robotic manipulation tasks, but their massive parameter scale leads to high computational and storage costs, posing deployment challenges. Existing acceleration approaches either rely on lightweight architectures that require costly retraining and exhibit limited generalization, or focus solely on inference acceleration without effectively reducing model size. To address these limitations, we propose DWIP, a depth–width integrated pruning strategy for VLA models. This method removes redundant layers from VLA models through dynamic layer pruning or task-driven adaptive layer pruning, enhancing inference efficiency while maintaining model performance. By further integrating width pruning, it constructs lightweight models that balance parameter size, accuracy, and speed under high pruning rates. Experimental results on the LIBERO benchmark demonstrate that DWIP enables lossless acceleration on simple tasks, and on complex tasks, achieves a 42.7% reduction in model parameters and a 1.82× inference speedup, while preserving 98.1% of the original model performance. Code is available at https://github.com/tlomlzq/DWIP.**

***Index Terms*—Vision-Language-Action models, pruning, inference acceleration, model compression**

I. Introduction

Vision-Language-Action (VLA) models [1]–[3] provide a new paradigm for robotic operations by unifying visual perception, language understanding, and action generation. However, existing VLA models often contain billions of parameters. Their massive scale not only imposes extremely high computational and storage costs during inference but also makes direct deployment on embodied agents or embedded chips challenging. Moreover, traditional autoregressive VLA models typically achieve inference rates of only 3–5 Hz [2], far below the millisecond-level responsiveness required for real-time robot interactions. These bottlenecks severely limit the application of VLA models in real robotic systems.

Existing research on efficient VLA models can be broadly categorized into two categories. The first focuses on redesigning network architectures [4]–[6]. Although these methods can improve inference efficiency, they require retraining, which incurs high computational costs [7], [8]. Moreover, the lack of large-scale training data for robots limits the models' generalization capacity. The second category focuses on optimizing inference efficiency by reducing computational overhead [9]–[11]. However, these methods fail to decrease the model's parameter size and storage requirements.

TABLE I
Modular Inference Performance Comparison of OpenVLA-7B and Its Lightweight Variants.

Model	Module	Parameters (M)	Inference Latency (ms)
OpenVLA-7B	Visual Enconder + Projector	802.29	26.6
	Language module (32 layers)	6738.94	311.7
Lightweight OpenVLA	Visual Enconder + Projector	802.29	25.1
	Language module (16 layers)	3500.81	149.7

To efficiently construct lightweight models, we systematically analyzed traditional VLA models based on autoregressive decoding. As shown in Table I, the OpenVLA-7B model [2] comprises a visual encoder, a projector, and a language module. The language module alone contributes roughly 90% of the model's parameters and inference latency. Notably, this module is a large language model (LLM) composed of multiple Transformer layers [12], which exhibit substantial structural redundancy, enabling the removal of certain layers or components with limited performance loss [13]–[15].

However, existing pruning methods for VLA models mainly focus on a single structural dimension. RLRC [16] compresses VLA models through width pruning and employs reinforcement learning to recover performance, but its additional interactive training incurs high computational and time costs. EfficientVLA [17] applies depth pruning to the language module and further accelerates inference by filtering visual tokens.

* Equal contribution. † Corresponding author.

However, its pruning ratio is relatively conservative, resulting in limited model compression and inference acceleration. This makes it difficult to meet the practical demands for highly efficient deployment on resource-constrained edge devices.

Further experimental analysis reveals that aggressive pruning in a single structural dimension not only disrupts the hierarchical and representational structure of the VLA models but also directly impairs the model's action generation capability. Since the outputs of VLA models are closely coupled with robotic arm actions, the removal of critical layers can lead to unstable control behaviors and task failures (see Sections III and V-C for details). This effect becomes particularly pronounced under high pruning ratios, where model performance is difficult to recover even with conventional fine-tuning.

Analysis of the root cause reveals that redundancy in VLA models is distributed across both depth and width structural spaces. Single-dimensional pruning at high pruning ratios tends to excessively disrupt critical computational pathways, thereby limiting the model's compressibility and recoverability under a fixed parameter scales. Based on this, this paper combines depth and width pruning strategies under preset parameter scale constraints to achieve effective model compression and inference acceleration while preserving action generation capabilities and key structural elements.

The contributions of this paper are summarized as follows:

- We find that two Transformer layers within the VLA model's language module are critical for performance in robotic manipulation tasks and must be preserved during pruning. Furthermore, different tasks have varying dependencies on layers.
- We propose DWIP, an integrated pruning strategy, which combines depth pruning and width pruning. This strategy effectively mitigates the performance degradation caused by depth pruning while reducing model size and accelerating inference, and exhibits strong robustness, especially in complex tasks.
- We propose a dynamic layer pruning method to efficiently remove redundant layers in language modules. Based on experimental results of skip layers, we further develop a task-driven adaptive layer pruning method.

II. Related work

A. Efficient Vision-Language-Action Models

Recent work on efficient VLA models has focused on two main aspects: model inference optimization and model architecture optimization.

a) Inference Optimization: PD-VLA [18] combines action chunking [19] with Jacobi decoding to enable parallel prediction of multiple tokens. To address the inefficient iterative process of Jacobi decoding, CEED-VLA [20] proposes an early-exit decoding strategy, further enhancing model inference speed. VLA-Cache [10] selectively reuses the computation results of visual tokens to reduce redundant computation. VLA-Pruner [21] improves computational efficiency by identifying critical visual tokens and pruning redundant tokens. DeeR-VLA [22] introduces a multi-exit inference framework that adaptively adjusts the effective model capacity based on task demands, thereby enabling early-exit computation. MoLe-VLA [11] dynamically selects the most relevant LLM layers based on the robot's current state to reduce computational costs. While these methods enhance inference efficiency, they do not reduce the model's parameter size or storage overhead.

b) Architecture Optimization: TinyVLA [5] and Evo-1 [23] use small multimodal models as the backbone of their policy networks to reduce computational overhead. SmolVLA [6] employs a small pre-trained Vision-Language Model (VLM) [24] and an efficient design to reduce the training and deployment overhead. But these methods often require retraining and are constrained by limited training data, which restricts their generalization ability.

B. Efficient Model Compression via Pruning

Pruning is a classic model compression technique that reduces model size and computational cost by removing unimportant parameters or structures [25]. Existing pruning methods can be broadly categorized into depth pruning and width pruning.

a) Width Pruning: Width pruning is a structured pruning method that reduces parameters and computation by removing less important channels or neurons while preserving model performance. AMP [26] identifies redundancy by evaluating the actual contributions of attention heads and multilayer perceptrons to the model's residual flow. OWLed [27] performs non-uniform intra-layer pruning based on the distribution of outlier features within each layer. While these methods can effectively reduce model size, their improvement in inference speed is relatively limited.

b) Depth Pruning: Depth pruning reduces model depth by removing entire layers while maintaining the weight dimensions of the remaining layers unchanged. Its core lies in identifying and eliminating less important layers within the model. LLM-Streamline [28] measures layer importance by calculating the cosine similarity between inputs and outputs, then performs depth pruning based on this metric. Previous studies indicate depth pruning methods not only significantly accelerate model inference but also achieve performance comparable to width pruning methods when combined with a Low-Rank Adaptation (LoRA) [29] fine-tuning recovery phase [30].

III. Skipping Layers Experiment

To compress the VLA model, we used the traditional cosine similarity-based pruning method, removing layers with high input-output similarity from the language module, starting from the penultimate layer and progressively eliminating redundant layers. We found that once the depth pruning ratio exceeded a certain threshold without fine-tuning, the task success rate (TSR) dropped sharply and even reached zero, as shown in Figure 1. Moreover, in robotic control, the arm moved sluggishly, the gripper stayed closed, and actions were invalid. These results indicate that simple layer removal fails to preserve model expressiveness and highlights the need for fine-tuning after pruning.

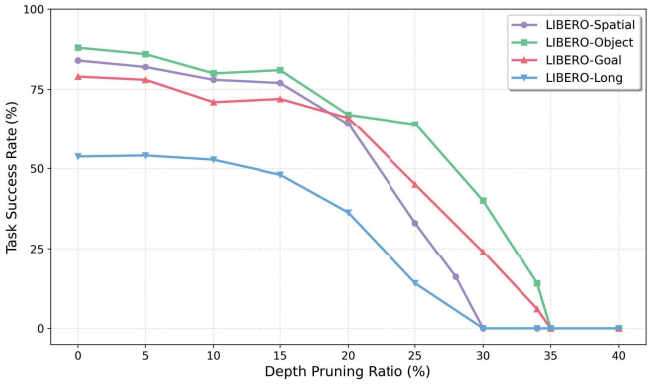

Fig. 1. The impact of depth pruning ratio on OpenVLA performance across different tasks (without fine-tuning; each task repeated 30 times).

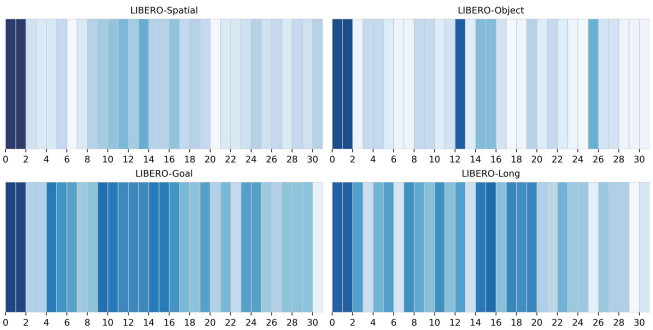

Fig. 2. Skipping Layer Experiment. The X-axis indicates the index of the skipped layer. The color intensity represents the deviation in task success rate after skipping the corresponding layer, with darker colors indicating higher layer importance.

To further identify redundant layers in the language module of the VLA models, we propose a simple experiment: replacing the output of a specific layer with its input to simulate skipping it, and measuring the impact on performance. Specifically, we use the OpenVLA checkpoints in the LIBERO benchmark [31] as base models and evaluate the influence of each skipped layer on different tasks without any further fine-tuning.

As shown in Figure 2, skipping the first two layers severely impairs performance. Furthermore, the final layer exhibits stronger semantic relevance to the downstream task of robotic manipulation [11]. Therefore, we designate the first two layers and the final layer as critical layers and maintain their integrity during pruning processes to avoid unnecessary degradation of the model capabilities.

IV. Method

A. Overview

Based on the experimental results in Section III, we propose DWIP, an integrated pruning strategy combining depth and width pruning. Its core idea is to jointly employ depth and width pruning methods under a target parameter constraint, balancing inference speed and model performance. Specifically, according to the results shown in Figure 1, when the depth pruning ratio exceeds approximately 30%, the model's performance on certain tasks drops sharply. Therefore, we set this ratio as the depth pruning threshold. Depth pruning is prioritized before significant performance degradation occurs to maximize the removal of redundant layers, effectively reducing inference latency. When the depth pruning ratio reaches the threshold, or the model has not yet met the preset size, width pruning is further applied for fine-grained compression. This mitigates structural damage caused by single-depth pruning while reducing computational and storage overhead.

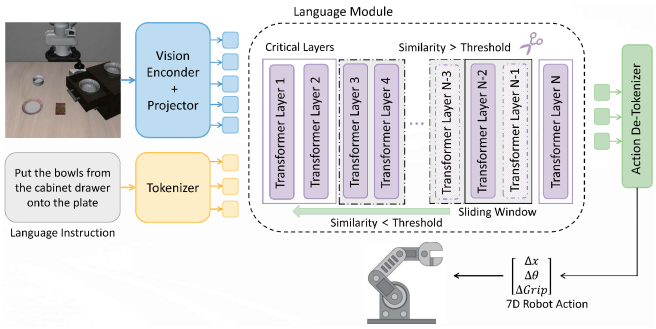

Fig. 3. The VLA model framework based on dynamic layer pruning.

B. Dynamic Layer Pruning

Unlike the pruning scenarios in LLMs that focus on text generation, the language module in VLA models serves as a critical intermediate representation between high-level task understanding and low-level action execution. The stability of its outputs directly impacts the generation of downstream continuous actions.

Based on this, in DWIP, we first apply a top-down dynamic layer pruning (DLP) strategy to the language module, using output similarity as a constraint during the pruning process. Specifically, as illustrated in Figure 3 , the DLP method employs a sliding window mechanism to dynamically evaluate the mergeability of adjacent layers. To ensure the stability of the pruned language module on key representations, we compute the mean cosine similarity of the last layer's hidden states on the representative task datasets $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$:

$$\bar{s} = \frac{1}{N} \sum_{i=1}^{N} \frac{h_L^{(i)} \cdot h_L^{(i)'}}{\|h_L^{(i)}\| \, \|h_L^{(i)'}\|}, \tag{1}$$

where $h_L^{(i)}$ and $h_L^{(i)'}$ denote the final-layer hidden state vectors of the original and the pruned model for sample i, respectively. N is the number of samples drawn from the corresponding task dataset. If the following condition is satisfied:

$$\bar{s} \geq \tau, \tag{2}$$

where τ is a predefined similarity threshold, the merge operation is considered acceptable, as it does not lead to significant degradation in model performance, and the merged result is retained. Otherwise, the operation is rejected, and the

sliding window is shifted downward from its current high-level position to continue evaluating the mergeability of the next set of adjacent layers. Finally, the original language module is replaced with the compressed language module to obtain the pruned VLA model.

The merging strategy uses weighted differential accumulation. Suppose the adjacent layers to be merged are layer $l-1$ and layer l, with respective weights W_{l-1} and W_l. The weights of the merged lower layers are updated as follows:

$$\widetilde{W}_{l-1} = W_{l-1} + \alpha \left(W_l - W_{l-1} \right), \tag{3}$$

where $\alpha \in [0, 1]$ is the weight fusion coefficient. Then, high-level structures are removed, thereby reducing redundant layers while maximizing the model's representational power. This process is iterated until the preset pruning ratio is achieved.

C. Width Pruning

For width pruning, we employ a structured pruning strategy [32] that compresses the coupled structures within the Transformer's Multi-Head Attention (MHA) and Multilayer Perceptron (MLP) modules. To ensure the independence of our work, only a brief introduction is provided here.

This method employs a first-order Taylor expansion to estimate the importance of the weight gradients of each coupled structure and ranks them within each layer. Subsequently, the importance of each layer is evaluated based on the cosine similarity of the Transformer layer's input and output, and non-uniform sparsity is allocated across different layers. Layers with higher similarity are assigned higher pruning ratios, while critical layers (the first two layers and the last layer) are protected and not assigned a pruning ratio to avoid performance degradation. Finally, under the allocated layer-wise sparsity constraints, less important parameter groups are progressively removed according to the local ranking, achieving intra-layer compression.

V. Experiments

A. Experimental Settings

a) Evaluation Environment: To evaluate the effectiveness of the proposed pruning strategy, we systematically tested the pruned models on the LIBERO simulation benchmark. This benchmark includes four task suites: Spatial, Object, Goal, and Long, which respectively assess the model's performance in spatial relationship understanding, object property modeling, goal-conditioned task execution, and long-horizon multi-stage task planning. Each task suite contains multiple robot manipulation tasks, each designed to evaluate the model's performance under a specific variation of the corresponding task type. Experiments cover all task suites to comprehensively evaluate the generalization ability of the pruned model under different scenarios and task conditions.

b) Baselines: We use the OpenVLA model as the baseline. Furthermore, based on observations from the layer-skipping experiments in Section III, we implemented a task-driven adaptive layer pruning method (TALP), which progressively removes layers with lesser impact on task success rates.

TABLE II
Performance comparison of pruning methods on OpenVLA on the LIBERO benchmark.

Method	Pruning Ratios	Spatial	Object	Goal	Long	Avg.
Original	-	84.8%	88.0%	76.6%	52.0%	75.4%
DLP	30%	83.2%	86.0%	73.6%	49.6%	73.1%
	50%	82.6%	85.4%	71.0%	47.4%	71.6%
	70%	81.8%	84.2%	69.4%	46.0%	70.4%
TALP	30%	**85.6%**	**89.0%**	72.4%	48.8%	74.0%
	50%	84.2%	86.2%	71.4%	46.4%	72.1%
	70%	83.0%	85.2%	69.4%	44.6%	70.6%
WP	30%	84.2%	88.0%	**76.4%**	**52.0%**	**75.2%**
	50%	83.0%	85.6%	71.4%	49.8%	72.5%
	70%	82.0%	84.2%	69.6%	47.6%	70.9%

B. Implementation Details

We applied different pruning methods to the benchmark model OpenVLA-7B and evaluated their performance on four task suites of LIBERO. Each task suite comprised 10 tasks, with each task repeated 50 times. The average task success rate served as the evaluation metric. All experiments were conducted on NVIDIA RTX 4090 and A800 GPUs. We implemented our pruning methods in PyTorch using the HuggingFace Transformers library.

During the iterative pruning process with dynamic layer pruning, we randomly selected 30 samples from the expert demonstration trajectories corresponding to each task suite in LIBERO as calibration data. This data was used to compute the cosine similarity between model outputs before and after pruning, as well as to evaluate the importance of parameter groups in the width pruning method. Unlike text-based calibration approaches, this method more effectively constrains the behavior of pruned models, enabling them to better adapt to robotic manipulation task scenarios.

Furthermore, to restore the performance of the pruned model, we employed LoRA [29] for efficient fine-tuning. Specifically, a low-rank matrix of rank 32 was inserted into each linear layer to achieve efficient parameter adaptation. We used two NVIDIA A800 GPUs with 80GB of memory for fine-tuning. For the LIBERO-Long suite, we fine-tuned for 50,000 steps, and for the other three suites for 20,000 steps, with a batch size of 16. In practice, we observed that models with lower pruning rates could recover performance with fewer fine-tuning steps, and that longer fine-tuning steps resulted in better performance recovery.

C. Evaluation Results on LIBERO

We systematically evaluated the proposed pruning strategies on the LIBERO benchmark under three relatively high pruning ratios. The results are shown in the Table II. Experimental results show that, across different pruning ratios, all pruning strategies maintain relatively high task success rates on the four tasks. This suggests that the VLA model contains substantial redundancy across both depth and width dimensions.

At a lower pruning ratio of 30%, TALP exhibits a certain advantage on simple tasks, with the task success rate increas-

TABLE III
EFFICIENCY ANALYSIS OF DEPTH AND WIDTH PRUNING METHODS UNDER DIFFERENT PRUNING RATIOS.

Pruning Ratios	Method	Params (B)	Inference Latency(ms)	Throughput (Hz)	FLOPs (10^{12})	Speed up
-	Original	7.5B	338.3	2.96	1.39	1.00×
30%	Width Pruning	5.8B	267.1	3.74	1.10	1.27×
50%		4.6B	222.9	4.49	0.90	1.52×
70%		3.5B	186.1	5.37	0.71	1.82×
30%	Depth Pruning	5.6B	233.8	4.28	0.96	1.45×
50%		4.3B	174.8	5.72	0.70	1.94×
70%		**2.9B**	**109.6**	**9.12**	**0.39**	**3.09**×

TABLE IV
PERFORMANCE COMPARISON OF INTEGRATED METHODS ON THE LIBERO-LONG SUITE. LATENCY IS TESTED ON 4090.

Method	Params	Task Success Rate	Inference Latency (ms)
Original-OpenVLA	7.5B	52.0%	338.3
TALP (50%)	4.3B	46.4% (-5.6%)	176.5 (↓47.8%)
DLP (50%)	4.3B	47.4% (-4.6%)	174.8 (↓48.3%)
WP (55%)	4.3B	49.4% (-2.6%)	218.7 (↓35.4%)
DWIP			
TALP (30%) + DLP (30%)	4.3B	47.7% (-4.3%)	**173.3 (↓48.8%)**
TALP (30%) + WP (32%)	4.3B	46.8% (-5.2%)	186.9 (↓44.7%)
DLP (30%) + WP (32%)	4.3B	**51.0% (-1.0%)**	185.7 (↓45.1%)
TALP (15%) + DLP (15%) + WP (32%)	4.3B	48.8% (-3.2%)	186.0 (↓45.0%)

ing from 84.8% to **85.6%**. This further validates the existence of redundancy layers that can be safely removed within the language module and also demonstrates the effectiveness of using TSR as an indicator for identifying redundant layers in this scenario. However, at the same pruning rate, DLP's performance slightly declined, indicating its advantages do not extend to low-complexity task scenarios.

As task complexity increases, DLP consistently outperforms TALP's direct layer removal strategy across varying pruning rates in the Long task suite. This demonstrates that the layer merging mechanism based on weight differential accumulation retains complex reasoning capabilities more effectively. Experimental results also reveal that simple tasks like Spatial and Object exhibit lower dependency on model size and inference depth, whereas complex tasks such as Goal and Long still require deeper representational and reasoning capabilities.

Regarding width pruning, WP shows only a **0.2%** average performance drop at the 30% pruning rate. Because it removes redundant channels without disrupting the model's overall structure, it demonstrates greater stability than depth pruning at high pruning rates. Further analysis of robotic arm behavior reveals that width pruning offers significant advantages in action generation continuity and executability compared to depth pruning, which can cause control instability such as oscillations or stagnation.

D. Efficiency Analysis

We tested the average single-step inference time of models after depth and width pruning at different pruning rates on an NVIDIA RTX 4090. Furthermore, we estimated the number of forward propagation floating-point operations for the attention layer and MLP layer of the language module, with a uniform sequence length of 128 tokens.

As shown in Table III, width pruning achieved acceleration factors of 1.27×, 1.52×, and 1.82× at pruning rates of 30%, 50%, and 70%, respectively. This is because width pruning reduces computational load by compressing the channels and MLP dimensions within a single layer while maintaining the total number of layers. By compressing single-layer channels and MLP dimensions while maintaining the same number of layers, width pruning reduces computational load and enhances inference speed.

Furthermore, due to the presence of protected layers, models obtained via width pruning retain a slightly higher number of parameters than those produced by depth pruning under the same pruning ratio.

Depth pruning significantly reduces forward computation by removing unimportant Transformer layers, thereby decreasing memory access between activations and weights. At the same pruning rates, depth pruning achieved acceleration factors of 1.45×, 1.94×, and 3.09×, demonstrating more pronounced performance gains compared to width pruning.

Although width and depth pruning operate on different principles, both effectively reduce computational complexity and parameter size, improving inference efficiency, with depth pruning exhibiting greater advantages at higher pruning ratios.

E. Ablation Study

Based on the experimental results presented above, we combined depth and width pruning within a fixed parameter scale to verify the effectiveness of DWIP at high pruning rates. We designed a series of experiments on the LIBERO-Long suite, with the experimental results shown in Table IV.

Specifically, the integrated pruning is conducted in two stages: depth pruning (TALP or DLP) is first applied to the original model, and width pruning is then performed on the compressed model obtained from the first stage. All integrated methods follow this stepwise procedure.

Experimental results demonstrate that combining depth pruning with width pruning outperforms using depth pruning alone, validating the effectiveness of DWIP at high pruning rates. The DLP + WP combination achieved the highest task success rate at high pruning rates, with only a **1%** decrease compared to the baseline, surpassing any single pruning strategy. This indicates that after removing redundant layers, VLA models retain compressible space in the width dimension. However, standalone width pruning tends to damage critical structures at high pruning rates, making it difficult to meet high compression ratio demands. DWIP preserves key model structures while effectively maintaining action generation capabilities and performance on complex tasks.

In contrast, TALP + WP shows limited improvement, likely because its discontinuous layer removal disrupts representation

continuity across layers at high pruning rates, weakening its ability to model long-term dependencies and complex decision relationships. Even when DLP is applied to mitigate this effect, the compensation afforded by its linear approximation is limited due to the irreversible structural damage, resulting in TALP + DLP + WP still underperforming DLP + WP.

VI. Conclusion

In this paper, we propose DWIP, an integrated pruning strategy based on depth–width integrated pruning to construct lightweight models that balance accuracy, speed, and parameter size. Our study demonstrates that pruning large-scale pretrained VLA models is an efficient and low-cost approach for model lightweighting. Furthermore, we find that the first two layers of the language module are critical for model performance, as skipping any layer across different tasks leads to significant performance degradation. To address this, we propose dynamic layer pruning and task-driven adaptive depth pruning methods, combined with width pruning to achieve multi-dimensional compression. This enables the model to maintain key structural integrity and task performance under high pruning ratios, providing a viable solution for deploying efficient VLA models.

References

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, vol. 229. PMLR, 2023, pp. 2165–2183.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong *et al.*, "Openvla: An open-source vision-language-action model," in *Conference on Robot Learning*, vol. 270. PMLR, 2024, pp. 2679–2713.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[4] J. Liu, M. Liu, Z. Wang, P. An, X. Li, K. Zhou, S. Yang, R. Zhang, Y. Guo, and S. Zhang, "Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 40 085–40 110, 2024.

[5] J. Wen, Y. Zhu, J. Li, M. Zhu, Z. Tang, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen *et al.*, "Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation," *IEEE Robotics and Automation Letters*, 2025.

[6] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti *et al.*, "Smolvla: A vision-language-action model for affordable and efficient robotics," *arXiv preprint arXiv:2506.01844*, 2025.

[7] S. Li, Z. Lai, D. Li, Y. Hao, W. Liu, K. Ge, X. Deng, and K. Lu, "Oases: Efficient large-scale model training on commodity servers via overlapped and automated tensor model parallelism," *IEEE Transactions on Parallel & Distributed Systems*, vol. 36, pp. 1828–1840, 2025.

[8] P. Liang, Y. Tang, X. Zhang, Y. Bai, T. Su, Z. Lai, L. Qiao, and D. Li, "A survey on auto-parallelism of large-scale deep learning training," *IEEE Transactions on Parallel & Distributed Systems*, vol. 34, pp. 2377–2390, 2023.

[9] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine, "Fast: Efficient action tokenization for vision-language-action models," *arXiv preprint arXiv:2501.09747*, 2025.

[10] S. Xu, Y. Wang, C. Xia, D. Zhu, T. Huang, and C. Xu, "Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation," *arXiv preprint arXiv:2502.02175*, 2025.

[11] R. Zhang, M. Dong, Y. Zhang, L. Heng, X. Chi, G. Dai, L. Du, Y. Du, and S. Zhang, "Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation," *arXiv preprint arXiv:2503.20384*, 2025.

[12] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Advances in neural information processing systems*, vol. 34, pp. 15 908–15 919, 2021.

[13] J. Xiao, P. Li, J. Nie, and Z. Tang, "Seven: Pruning transformer model by reserving sentinels," in *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–8.

[14] S. Yuan, E. Nie, B. Ma, and M. Färber, "Why lift so heavy? slimming large language models by cutting off the layers," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–8.

[15] G. Yao, Y. Wang, W. Zhang, X. Deng, C. Du, R. Han, Z. Qin, Y. Li, B. Xie, B. Dong *et al.*, "Rl-pruner: Retraining-free global exploration pruning method based on reinforcement learning," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–8.

[16] Y. Chen and X. Li, "Rlrc: Reinforcement learning-based recovery for compressed vision-language-action models," *arXiv preprint arXiv:2506.17639*, 2025.

[17] Y. Yang, Y. Wang, Z. Wen, L. Zhongwei, C. Zou, Z. Zhang, C. Wen, and L. Zhang, "Efficientvla: Training-free acceleration and compression for vision-language-action models," *arXiv preprint arXiv:2506.10100*, 2025.

[18] W. Song, J. Chen, P. Ding, H. Zhao, W. Zhao, Z. Zhong, Z. Ge, J. Ma, and H. Li, "Accelerating vision-language-action model integrated with action chunking via parallel decoding," *arXiv preprint arXiv:2503.02310*, 2025.

[19] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[20] W. Song, J. Chen, P. Ding, Y. Huang, H. Zhao, D. Wang, and H. Li, "Ceed-vla: Consistency vision-language-action model with early-exit decoding," *arXiv preprint arXiv:2506.13725*, 2025.

[21] Z. Liu, Y. Chen, H. Cai, T. Lin, S. Yang, Z. Liu, and B. Zhao, "Vla-pruner: Temporal-aware dual-level visual token pruning for efficient vision-language-action inference," *arXiv preprint arXiv:2511.16449*, 2025.

[22] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang, "Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution," *Advances in Neural Information Processing Systems*, vol. 37, pp. 56 619–56 643, December 2024.

[23] T. Lin, Y. Zhong, Y. Du, J. Zhang, J. Liu, Y. Chen, E. Gu, Z. Liu, H. Cai, Y. Zou *et al.*, "Evo-1: Lightweight vision-language-action model with preserved semantic alignment," *arXiv preprint arXiv:2511.04555*, 2025.

[24] A. Marafioti, O. Zohar, M. Farré, M. Noyan, E. Bakouch, P. Cuenca, C. Zakka, L. B. Allal, A. Lozhkov, N. Tazi *et al.*, "Smolvlm: Redefining small and efficient multimodal models," *arXiv preprint arXiv:2504.05299*, 2025.

[25] Y. Xue, W. Yao, S. Peng, S. Yang, and K. Zhou, "One-shot hierarchical global pruning via channel relation-aware attention," in *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–8.

[26] L. G. Mugnaini, B. Lopes Yamamoto, L. L. de Alcantara, V. Zacarias, E. Bollis, L. Pellicer, A. H. Reali Costa, and A. Jordao, "Efficient llms with amp: Attention heads and mlp pruning," in *2025 International Joint Conference on Neural Networks (IJCNN)*, 2025, pp. 1–8.

[27] J. Li, L. Yin, and X. Wang, "Owled: Outlier-weighed layerwise pruning for efficient autonomous driving framework," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–8.

[28] X. Chen, Y. Hu, J. Zhang, Y. Wang, C. Li, and H. Chen, "Streamlining redundant layers to compress large language models," in *The Thirteenth International Conference on Learning Representations*, 2025.

[29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[30] B.-K. Kim, G. Kim, T.-H. Kim, T. Castells, S. Choi, J. Shin, and H.-K. Song, "Shortened llama: A simple depth pruning for large language models," in *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[31] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44 776–44 791, 2023.

[32] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," *Advances in neural information processing systems*, vol. 36, pp. 21 702–21 720, 2023.