DMQA Open Seminar

Advanced Vision-Language-Action Models (Part 2): Training-Free Approaches for Efficient VLA

2026.11.06

고려대학교 산업경영공학과

Data Mining & Quality Analytics Lab.

이정민

KOREA UNIVERSITY

DMQA

발표자 소개



❖ 이정민(JungMin Lee)

- 고려대학교 산업경영공학과 대학원 재학
- Data Mining & Quality Analytics Lab.(김성범 교수님)
- 석박 통합 과정(2022.03~Present)

❖ Research Interest

- Uncertainty Quantification
- Label Noise Learning
- Vision-Language Models / Vision-Language-Action Models

❖ Contact

- jungmin9195@korea.ac.kr

Copyright ⓒ 2026, All rights reserved.

Contents

❖ **Introduction: What is Vision-Language-Action(VLA) Models?**

❖ **How: Training-Free Approaches for Efficient VLA Models**

- EfficientVLA (2026, NeurIPS)
- VLA-Cache (2026, NeurIPS)

❖ **Conclusion**

❖ **References**

Copyright © 2026, All rights reserved.

Introduction

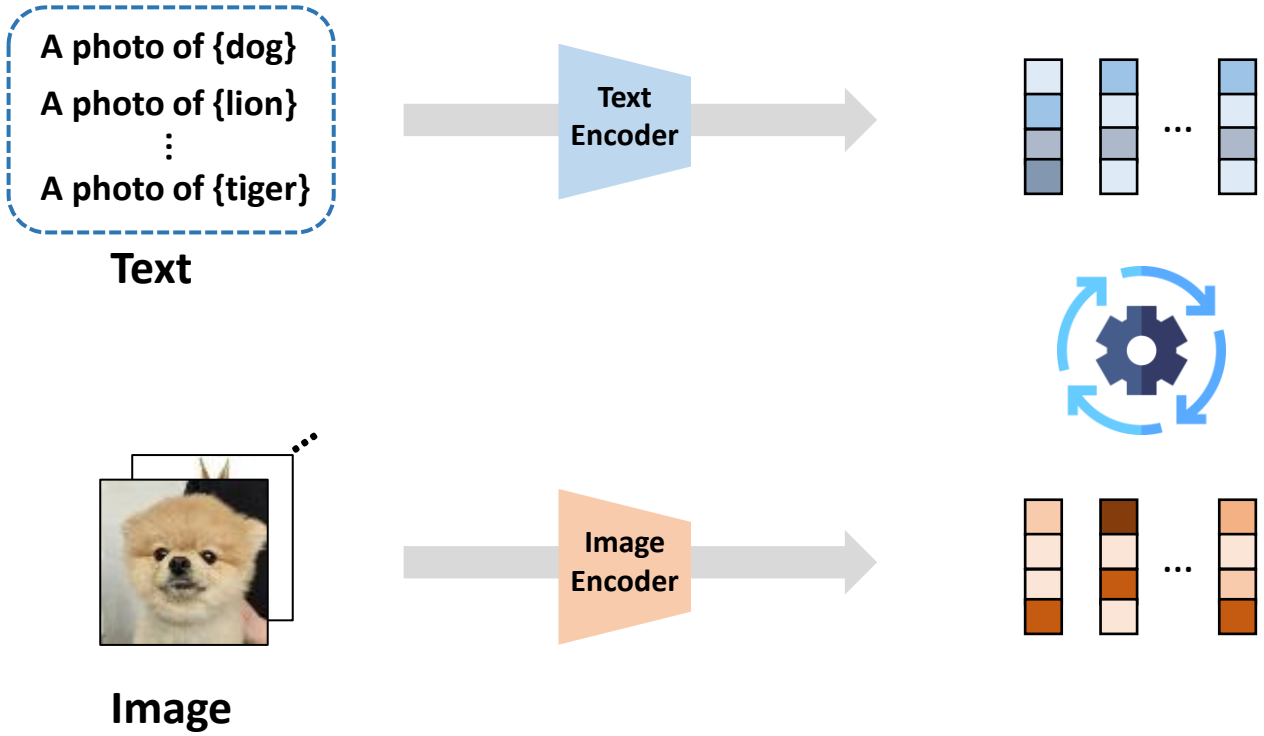
Copyright © 2023, All rights reserved.

Introduction

Vision-Language Models (VLM)

❖ **Vision-Language Models (VLM)란 무엇일까?**

- **시각(Vision)**과 **언어(Language)** 정보를 결합하여 사용할 수 있는 모델
- OOD detection, anomaly detection 등 다양한 task에 활발히 사용됨



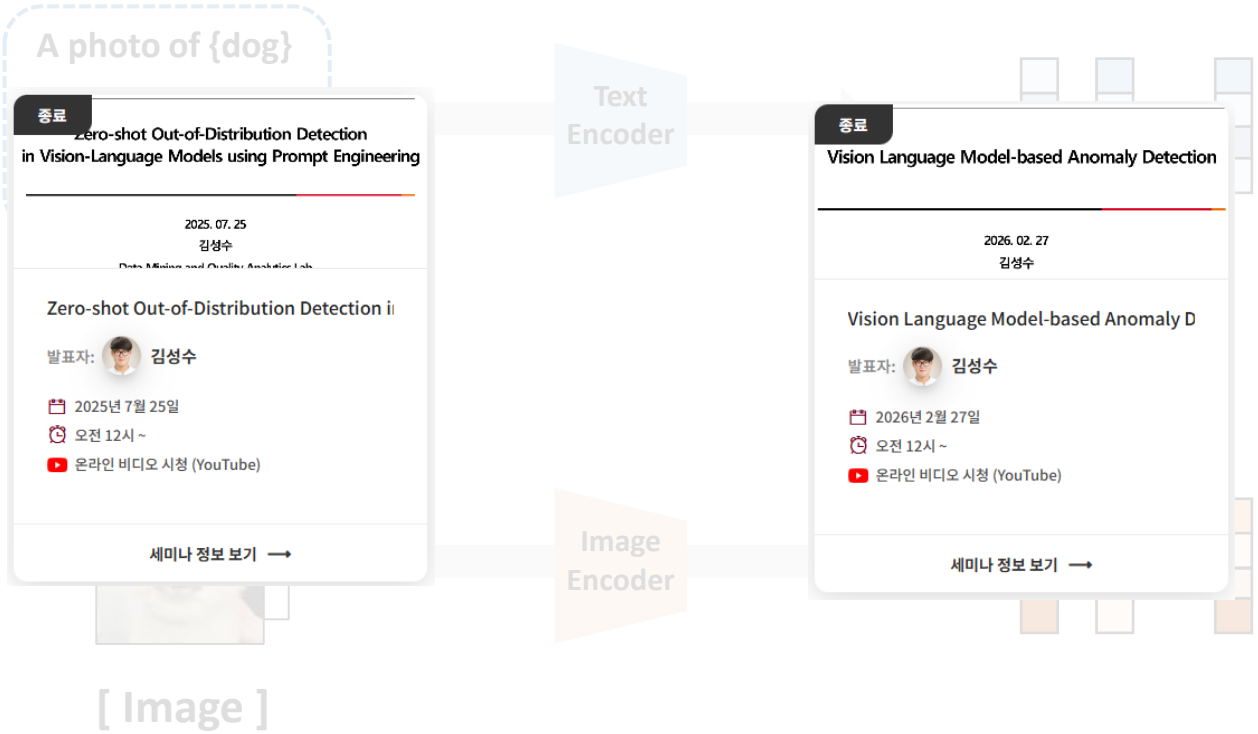
Copyright © 2026, All rights reserved.

Introduction

Vision-Language Models (VLM)

❖ Vision-Language Models (VLM)란 무엇일까?

- **시각(Vision)**과 **언어(Language)** 정보를 결합하여 사용할 수 있는 모델
- OOD detection, anomaly detection 등 다양한 task에 활발히 사용됨



Copyright © 2026, All rights reserved.

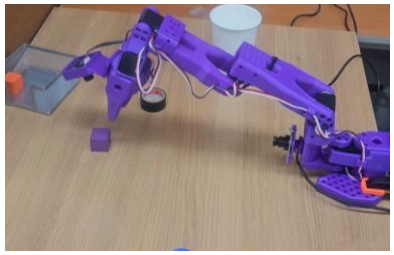
DMQA

Introduction

Vision-Language-Action (VLA) Models

❖ **Vision-Language-Action (VLA) Models란 무엇일까?**

- 사전학습된 VLM/MLLM을 기반으로 **시각**/**언어** 표현을 추출하고, action module을 통해 **로봇 제어 명령**을 예측



Vision(o_t)

Pick the cube and place it into the cup

Language (l_t)

VLM/MLLM

Action Module

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \dots \\ \Delta G \end{bmatrix}$$



Action (a_t)

Copyright © 2026, All rights reserved.

Introduction

Vision-Language-Action (VLA) Models

❖ **VLA을 어떻게 활용할 수 있을까?**

- 복잡한 협업 및 고난이도의 정밀한 task 수행



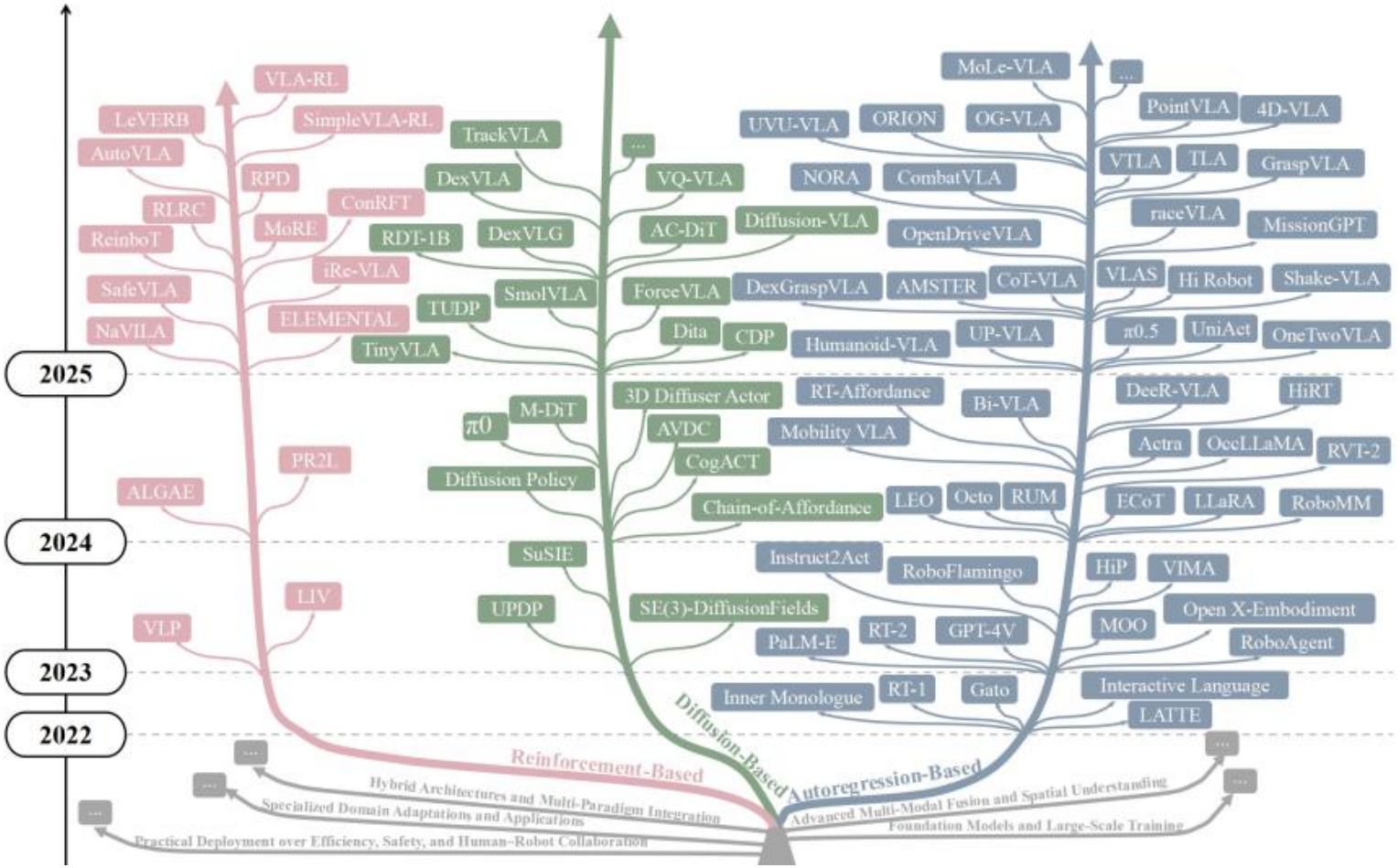
https://www.pi.website/blog/pistar06

Copyright © 2026, All rights reserved.

Introduction

Research Trends

❖ **VLA의 연구 동향은 어떻게 될까? (배경 방법론 관점)**



Zhang, D., Sun, J., Hu, C., Wu, X., Yuan, Z., Zhou, R., ... & Zhou, Q. (2025). Pure vision language action (vla) models: A comprehensive survey. arXiv preprint arXiv:2509.19012.

Copyright © 2026, All rights reserved.

DMQA

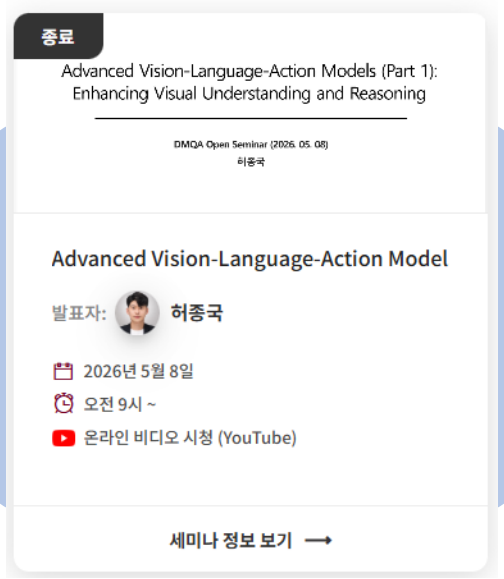
Introduction

Research Trends

❖ **VLA의 연구 동향은 어떻게 될까? (문제 상황 관점)**

- 장기 태스크 수행 및 forgetting 현상 해결을 위한 memory module 고도화
- 복잡한 상황에 대한 시각적 이해 및 구체적인 계획 수립을 위한 CoT 기반 reasoning
- 작업 속도를 고려한 경량화를 통해 VLA 모델의 효율성 개선

Memory for Long Horizon Tasks



Efficiency

Copyright © 2026, All rights reserved.

DMQA

Introduction

Research Trends

❖ **VLA의 연구 동향은 어떻게 될까? (문제 상황 관점)**

- 장기 태스크 수행 및 forgetting 현상 해결을 위한 memory module 고도화
- 복잡한 상황에 대한 시각적 이해 및 구체적인 계획 수립을 위한 CoT 기반 reasoning
- 작업 속도를 고려한 경량화를 통해 VLA 모델의 효율성 개선

Memory for Long Horizon Tasks

Reasoning for Complex-Decision Making

Efficiency

Copyright © 2026, All rights reserved.

Training-Free Approaches for Efficient VLA Models

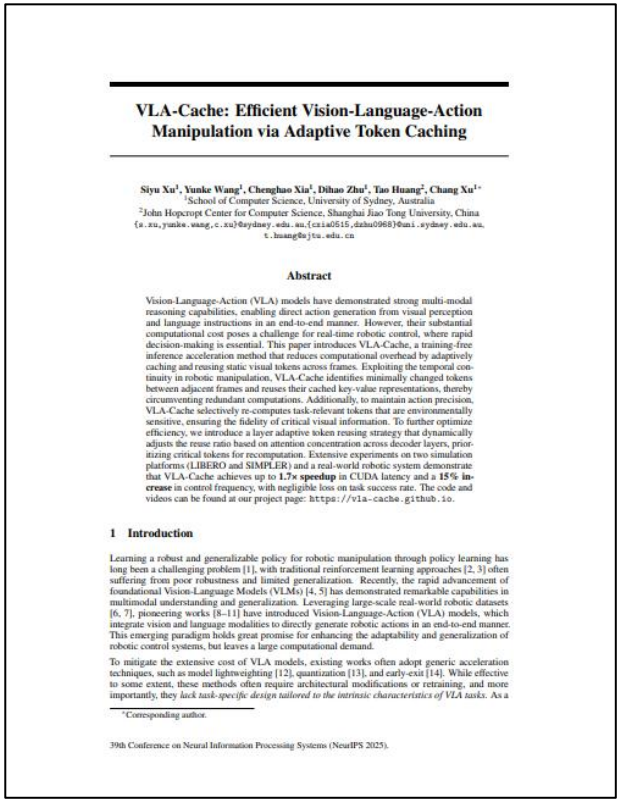
Copyright © 2023, All rights reserved.

VLA-Cache

Paper

❖ **VLA-Cache: Efficient Vision-Language-Action Manipulation via Adaptive Token Caching (2026, NeruIPS)**

- 시점에 따라 변하지 않는 visual token의 정보는 업데이트하지 않고 재사용하는 방법론 제안



VLA-Cache: Efficient Vision-Language-Action Manipulation via Adaptive Token Caching

Siyu Xu[1], Yunke Wang[1], Chenghao Xia[1], Dihao Zhu[1], Tao Huang[2], Chang Xu[1]*
[1]School of Computer Science, University of Sydney, Australia
[2]John Hopcropt Center for Computer Science, Shanghai Jiao Tong University, China
{s.xu,yunke.wang,c.xu}@sydney.edu.au,{czia0515,dzhu0968}@uni.sydney.edu.au,
t.huang@sjtu.edu.cn

Abstract

Vision-Language-Action (VLA) models have demonstrated strong multi-modal reasoning capabilities, enabling direct action generation from visual perception and language instructions in an end-to-end manner. However, their substantial computational cost poses a challenge for real-time robotic control, where rapid decision-making is essential. This paper introduces VLA-Cache, a training-free inference acceleration method that reduces computational overhead by adaptively caching and reusing static visual tokens across frames. Exploiting the temporal continuity in robotic manipulation, VLA-Cache identifies minimally changed tokens between adjacent frames and reuses their cached key-value representations, thereby circumventing redundant computations. Additionally, to maintain action precision, VLA-Cache selectively re-computes task-relevant tokens that are environmentally sensitive, ensuring the fidelity of critical visual information. To further optimize efficiency, we introduce a layer adaptive token reusing strategy that dynamically adjusts the reuse ratio based on attention concentration across decoder layers, prioritizing critical tokens for recomputation. Extensive experiments on two simulation platforms (LIBERO and SIMPLER) and a real-world robotic system demonstrate that VLA-Cache achieves up to **1.7× speedup** in CUDA latency and a **15% increase** in control frequency, with negligible loss on task success rate. The code and videos can be found at our project page: https://vla-cache.github.io.

1 Introduction

Learning a robust and generalizable policy for robotic manipulation through policy learning has long been a challenging problem [1], with traditional reinforcement learning approaches [2, 3] often suffering from poor robustness and limited generalization. Recently, the rapid advancement of foundational Vision-Language Models (VLMs) [4, 5] has demonstrated remarkable capabilities in multimodal understanding and generalization. Leveraging large-scale real-world robotic datasets [6, 7], pioneering works [8–11] have introduced Vision-Language-Action (VLA) models, which integrate vision and language modalities to directly generate robotic actions in an end-to-end manner. This emerging paradigm holds great promise for enhancing the adaptability and generalization of robotic control systems, but leaves a large computational demand.

To mitigate the extensive cost of VLA models, existing works often adopt generic acceleration techniques, such as model lightweighting [12], quantization [13], and early-exit [14]. While effective to some extent, these methods often require architectural modifications or retraining, and more importantly, they *lack task-specific design tailored to the intrinsic characteristics of VLA tasks*. As a

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Xu, S., Wang, Y., Xia, C., Zhu, D., Huang, T., & Xu, C. (2026). Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. Advances in Neural Information Processing Systems, 38, 164448-164473.

Copyright © 2026, All rights reserved.

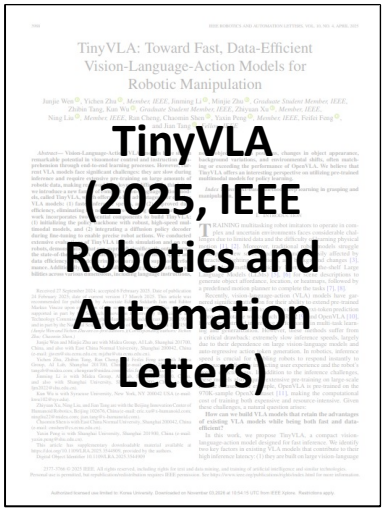
DMQA

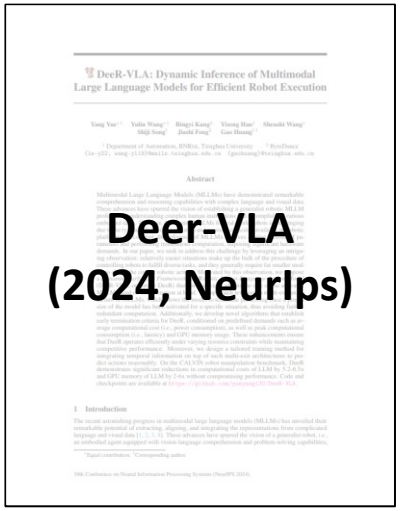
VLA-Cache

Introduction

❖ **기존 VLA efficiency 연구들은 어떤 한계를 가지고 있을까?**

- 모델을 구조적으로 수정
- Task-specific하게 적용하기 어려움





Copyright © 2026, All rights reserved.

DMQA

VLA-Cache

Introduction

❖ **VLA의 어떤 문제점을 지적하고 있을까?**

- 시점마다 크게 변하지 않는 시각 정보를 매번 업데이트 하는 것은 비효율적
  - ➢ 타겟 물체, 그리퍼 등의 정보는 시각적으로 크게 변하지 않아도 의미적으로 중요할 수 있음



[Previous Step (T-1)]



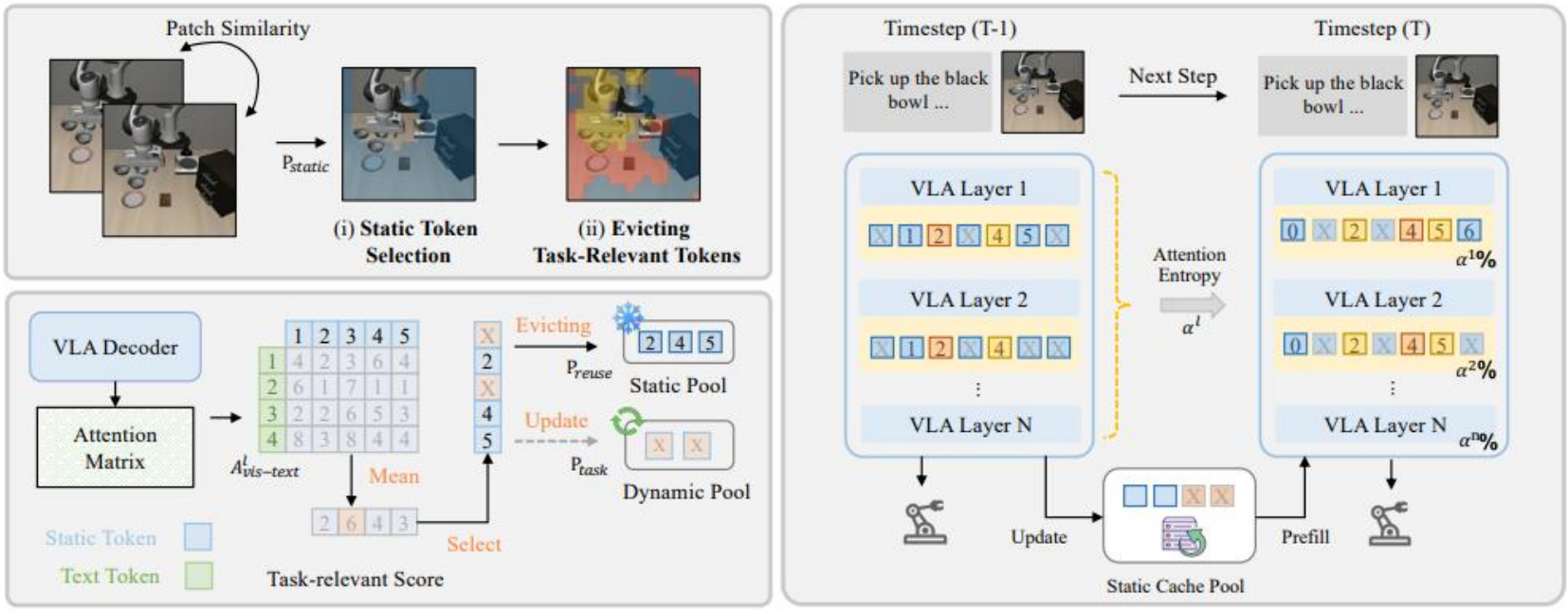
[Current Step (T)]

Copyright © 2026, All rights reserved.

VLA-Cache

Overall Framework

❖ **어떻게 VLA efficiency 문제를 해결하였을까?**



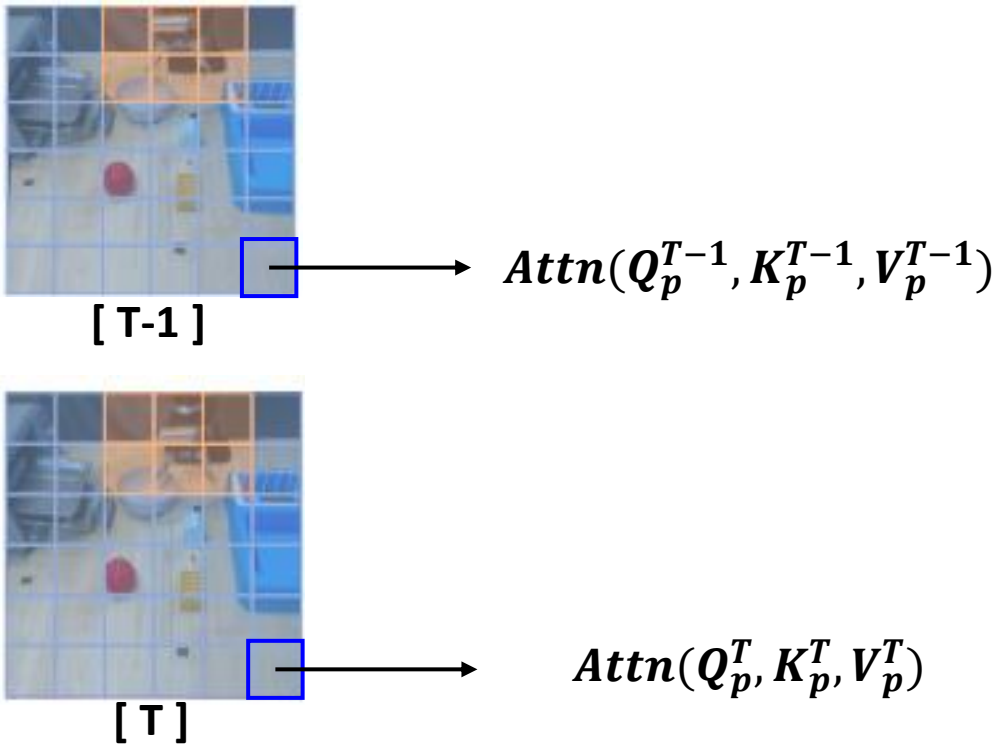
(a) Dynamic Token Selection (b) Adaptive Token Caching

Copyright © 2026, All rights reserved.

VLA-Cache

Method

❖ 어떤 측면으로 VLA efficiency 문제를 해결하였을까?



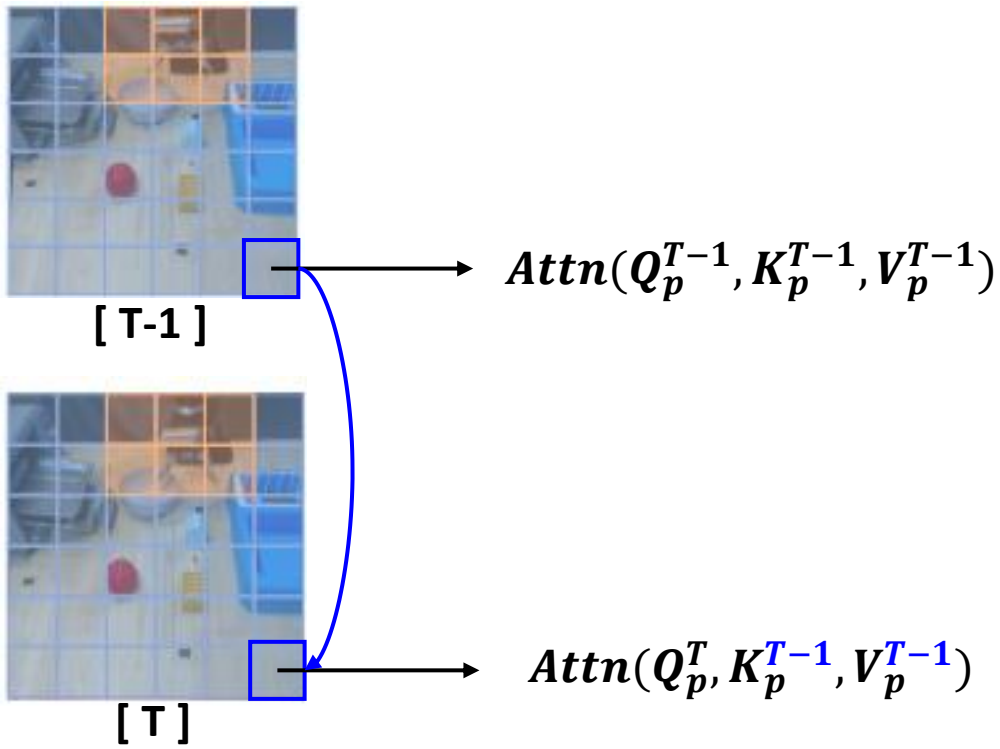
Copyright © 2026, All rights reserved.

VLA-Cache

Method

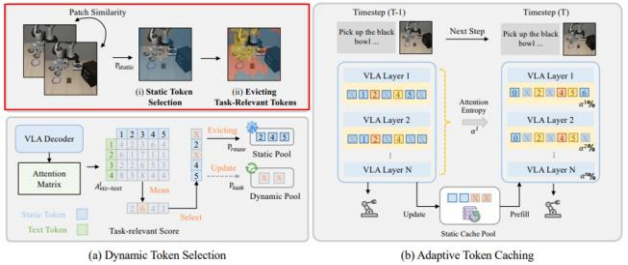
❖ **어떤 측면으로 VLA efficiency 문제를 해결하였을까?**

- Attention 과정에서 재사용 할 visual token (patch)의 **key, value 값은 이전 시점의 값을 사용**



Copyright © 2026, All rights reserved.

VLA-Cache

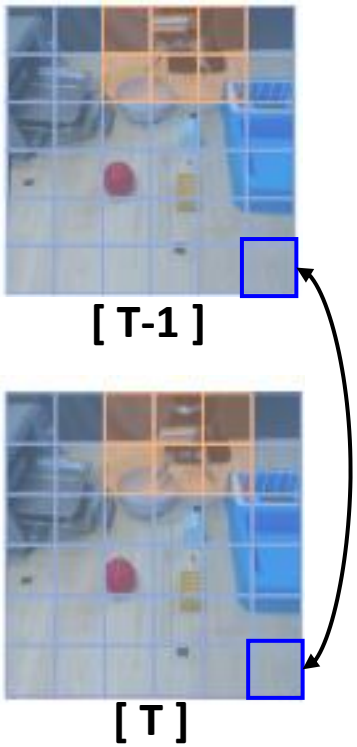


(a) Dynamic Token Selection (b) Adaptive Token Caching

Method

❖ **어떻게 재사용하는 static token 후보를 정의 하였을까?**

- Patch (token)별로 이전 시점과의 유사도를 계산하여 유사도가 높은 Top-k token들은 재사용 후보가 됨

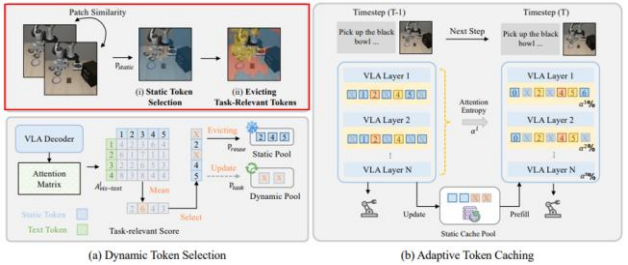


$$\mathbf{Sim}\left(\mathbf{P_t^{i,j}}, \mathbf{P_{t-1}^{i,j}}\right) = \frac{\mathbf{P_t^{i,j}} \cdot \mathbf{P_{t-1}^{i,j}}}{\left\|\mathbf{P_t^{i,j}}\right\|_2 \left\|\mathbf{P_{t-1}^{i,j}}\right\|_2}$$

Copyright © 2026, All rights reserved.

DMQA

VLA-Cache



(a) Dynamic Token Selection (b) Adaptive Token Caching

Method

❖ **Static token 전체를 재사용 하면 어떻게 될까?**

- Task-relevant token이 업데이트 되지 않아 성능이 대폭 하락함

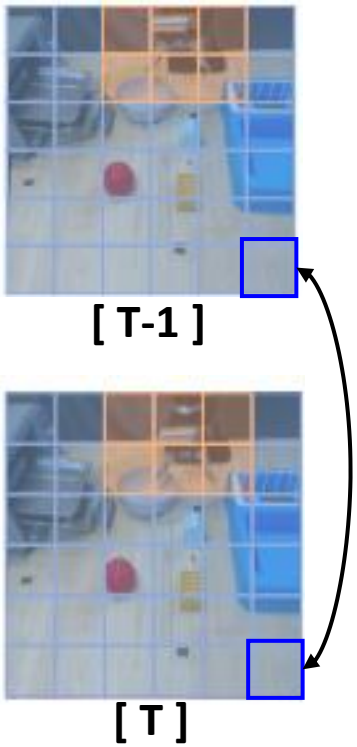


Table 1: Comparison of VLA-Cache's core token selection strategies on OpenVLA using the LIBERO Spatial benchmark.

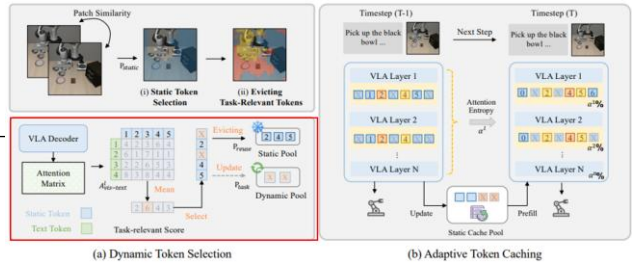
Method	SR (%) ↑	Latency (ms) ↓
OpenVLA	84.4	51.56
+ Static Token	74.2	31.03
+ Evict Task-Relevant	82.6	31.03
+ Layer Adaptive	83.8	32.22

$$\mathbf{Sim}\left(\mathbf{P_t^{i,j}}, \mathbf{P_{t-1}^{i,j}}\right) = \frac{\mathbf{P_t^{i,j}} \cdot \mathbf{P_{t-1}^{i,j}}}{\left\|\mathbf{P_t^{i,j}}\right\|_2 \left\|\mathbf{P_{t-1}^{i,j}}\right\|_2}$$

Copyright © 2026, All rights reserved.

DMQA

VLA-Cache



(a) Dynamic Token Selection (b) Adaptive Token Caching

Method

❖ **어떻게 task-relevant token을 정의 하였을까?**

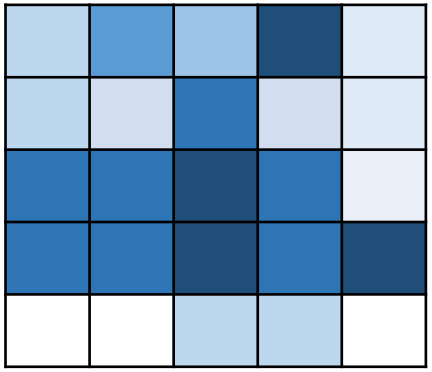
- Text prompt와의 **text-to-vision attention score**를 기반으로 threshold 이상의 token들을 task-relevant token으로 정의
- 모든 layer의 attention score를 평균하여 최종 attention score로 사용

[T-1]


[T]

$$P_{reuse} = P_{static} \setminus P_{task-relevant}$$

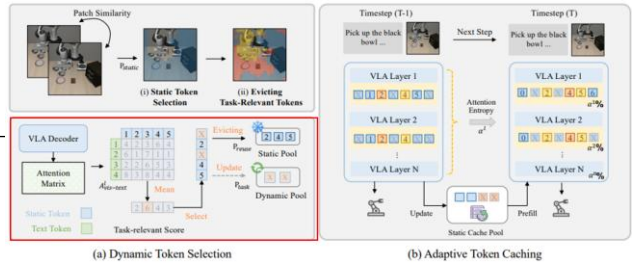
Pick up the gray bowl



Copyright © 2026, All rights reserved.

DMQA

VLA-Cache



(a) Dynamic Token Selection (b) Adaptive Token Caching

Method

❖ **어떻게 task-relevant token을 정의 하였을까?**

- Text prompt와의 **text-to-vision attention score**를 기반으로 threshold 이상의 token들을 task-relevant token으로 정의
- 모든 layer의 attention score를 평균하여 최종 attention score로 사용

[T-1]



[T]

$$P_{reuse} = P_{static} \backslash P_{task-relevant}$$

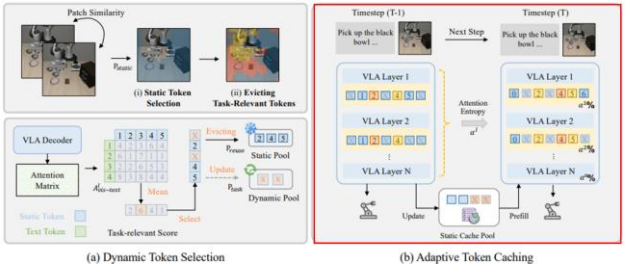
각 layer마다 가진 정보가 다름

Pick up the gray bowl

Copyright © 2026, All rights reserved.

DMQA

VLA-Cache



(a) Dynamic Token Selection (b) Adaptive Token Caching

Method

❖ **최종적으로 layer마다 어떻게 재사용 token을 정의 하였을까?**

- Entropy ϵ가 작을 수록 특정 token들에 attention이 집중되어 있음을 의미
- 누적 entropy 감소율이 클 수록 attention이 집중되지 않은 token들은 증가하기 때문에 재사용 비율 증가
- 재사용 비율 α^l 만큼 P_{reuse}에서 **이전 시점과의 유사도가 높은 token들을 최종 재사용 (key, value)**

[T-1]


[T]

$$R^l = \frac{\epsilon^{l-1} - \epsilon^l}{\epsilon^{l-1}}$$

$$\alpha^l = \min(k \sum_{j=1}^{l} R^j, 1)$$

Copyright © 2026, All rights reserved.

DMQA

VLA-Cache

Method

❖ 최종적으로 layer마다 어떻게 재사용 token을 정의 하였을까?

- Layer마다 entropy를 활용하여 재사용 비율 α^l을 정의
- Entropy가 높을수록 이전 시점과의 유사도가 낮은 token들이 많아 재사용 비율이 적음
- 재사용 비율만큼 token별로 이전 시점과의 유사도가 높은 token들을 최종 재사용 (key, value)

1. Token 별로 **이전 시점과의 유사도**를 계산하여 **Top-k static token** 정의

2. Static token 중 **text-to-vision attention score**를 기준으로 threshold를 넘는 token들은 **task-relevant token**으로 정의

3. Static token에서 task-relevant token을 제외하고 나머지를 layer마다 최종 재사용 후보로 정의

4. Layer마다 **entropy**를 활용하여 재사용 비율을 정의하고 그 비율만큼의 **상위 유사도를 가지는 token**은 재사용

$$R^l = \frac{\epsilon^{l-1}}{\epsilon^{l-1} - \epsilon^l}$$

$$\alpha^l = \min\left(k \sum_{j=1}^{l} R^j, 1\right)$$

[T-1]

[T]

Copyright © 2026, All rights reserved.

DMQA

VLA-Cache

Experiments

❖ **여러가지 환경에서 VLA-Cache의 성능은 어떨까?**

- 성능과 효율성을 동시에 개선

Table 2: Comparison of different VLA acceleration methods on the LIBERO benchmark.

Method	Success Rate ↑					FLOPs (T)↓	Latency (ms)↓	Control Freq. (Hz)↑
	Spatial	Object	Goal	Long	Average			
OpenVLA	84.4%	86.6%	75.6%	53.2%	75.0%	1.864	51.91	4.23
+ SparseVLM	79.8%	67.0%	72.6%	39.4%	64.7%	1.407	83.39	3.72
+ FastV	83.4%	84.0%	74.2%	51.6%	73.3%	1.864	53.28	4.19
+ VLA-Cache	**83.8%**	**85.8%**	**76.4%**	**52.8%**	**74.7%**	**1.355**	**31.83**	**4.59**
OpenVLA-OFT	97.8%	97.6%	97.6%	94.2%	96.8%	4.013	79.05	65.10
+ VLA-Cache	**98.3%**	97.5%	**98.3%**	**95.4%**	**97.4%**	**3.097**	**62.59**	**78.98**

Table 3: Comparison of VLA-Cache within the CogACT model in the SIMPLER environment.

SIMPLER	Method	Success Rate ↑					FLOPs (T)↓	Latency (ms) ↓	Control Freq. (Hz) ↑
		PickCan	MoveNear	Drawer	DrawerApple	Average			
Matching	CogACT	91.3%	**85.0%**	**71.8%**	50.9%	**74.8%**	1.847	54.29	12.42
	+ VLA-Cache	**92.0%**	83.3%	70.5%	**51.6%**	74.4%	**1.496**	**39.63**	**14.66**
Aggregation	CogACT	89.6%	80.8%	28.3%	**46.6%**	61.3%	1.807	53.54	12.36
	+ VLA-Cache	**91.7%**	**79.3%**	**32.5%**	45.8%	**62.3%**	**1.493**	**39.11**	**14.48**

Copyright © 2026, All rights reserved.

VLA-Cache

Experiments

❖ 실제로 VLA-Cache는 static token과 task-relevant token을 잘 구분할까?

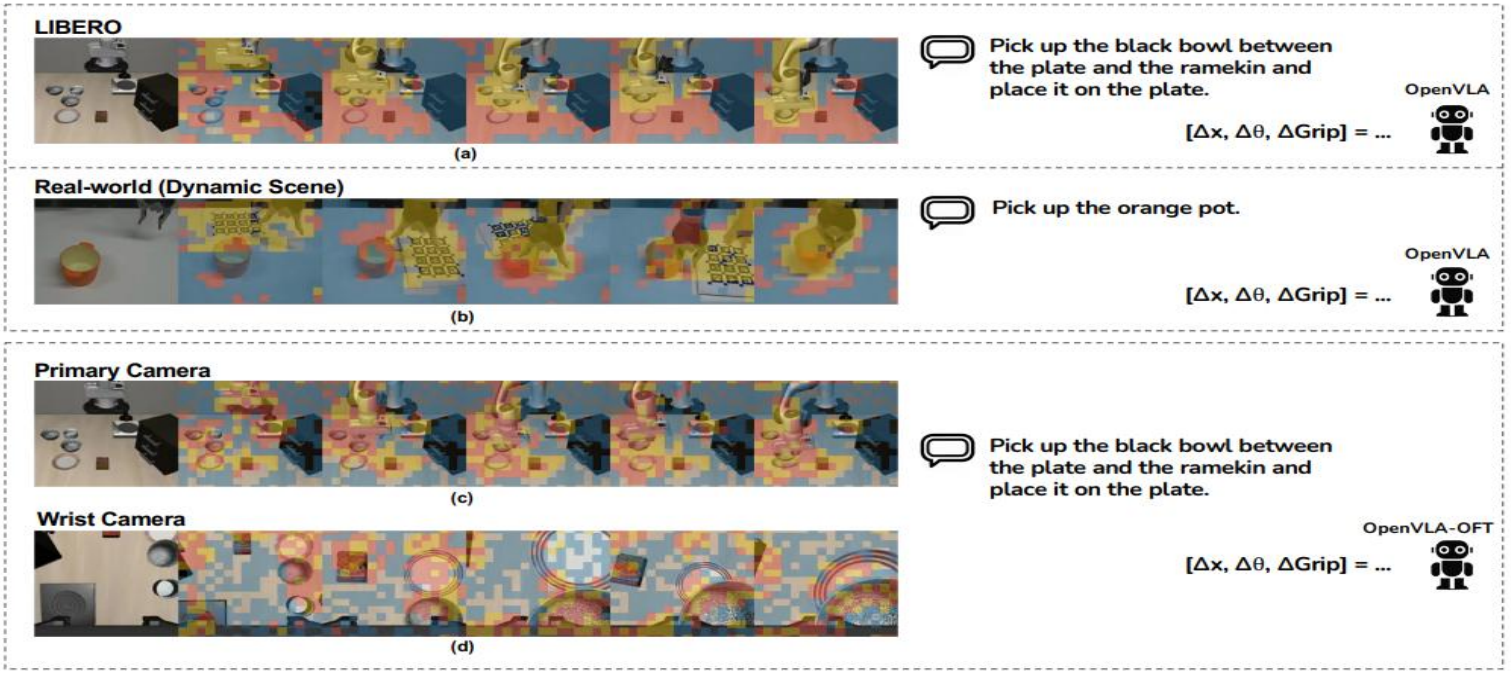


Figure 4: Visualization of VLA-Cache token reuse across settings. **(a)** LIBERO simulation with OpenVLA. **(b)** Real-world task under dynamic background. **(c)** and **(d)** Main and wrist camera views from OpenVLA-OFT. Blue: static tokens, Yellow: task-relevant, Red: overlapping. VLA-Cache reduces redundant computation and preserves accuracy under varying conditions.

Copyright © 2026, All rights reserved.

VLA-Cache

Experiments

❖ **실제 로봇 환경에서 VLA-Cache의 성능은 어떻게 될까?**

- 효율성 측면 외적에서 봤을 때, 모든 token을 매번 업데이트 하는 것이 항상 효과적이지는 않음을 보임

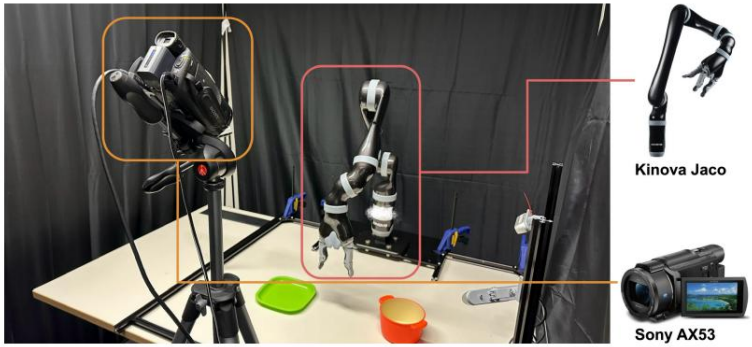


Table 5: Comparison of success rate on real robot tasks.

Method	Success Rate ↑					FLOPs (T)↓	Latency (ms) ↓	Control Freq. (Hz) ↑
	PickPot	PlaceCube	PutSausage	WipeTable	Average			
OpenVLA	**95.0%**	83.3%	80.0%	70.0%	82.1%	1.814	64.16	4.02
+ **VLA-Cache**	90.0%	**90.0%**	**85.0%**	**73.3%**	**84.6%**	**1.303**	**51.85**	**4.21**

Copyright © 2026, All rights reserved.

VLA-Cache

Experiments

❖ **Hyperparameter에 따라 VLA-Cache의 성능은 어떻게 될까?**

- 재사용되는 token이 많아져도 성능은 안정적이며 효율성은 점차 개선

Method	SR (%) ↑	FLOPs (T) ↓	Latency (ms) ↓	Freq. (Hz) ↑
OpenVLA-OFT	97.8	3.995	78.35	65.44
+ VLA-Cache (k=50)	97.6	3.332	66.82	77.23
+ VLA-Cache (k=80)	97.8	3.226	66.55	78.66
+ VLA-Cache (k=100)	**98.2**	3.156	64.88	79.88
+ VLA-Cache (k=120)	98.0	3.109	62.99	80.56
+ VLA-Cache (k=150)	97.4	**3.043**	**61.12**	**81.67**
+ VLA-Cache (k=180)	96.6	2.936	60.46	82.51

Table 9: Varying the static-token budget k (with τ=0.5).

Method	SR (%) ↑	FLOPs (T) ↓	Latency (ms) ↓	Freq. (Hz) ↑
OpenVLA-OFT	97.8	3.995	78.35	65.44
+ VLA-Cache (τ=0.2)	95.6	3.384	66.93	76.74
+ VLA-Cache (τ=0.3)	96.2	3.283	67.03	79.31
+ VLA-Cache (τ=0.4)	98.0	3.204	66.38	79.79
+ VLA-Cache (τ=0.5)	**98.2**	3.156	64.88	79.88
+ VLA-Cache (τ=0.6)	98.6	3.131	64.27	81.98
+ VLA-Cache (τ=0.7)	98.4	**3.068**	**63.12**	**82.96**

Table 10: Varying the relevance threshold τ (with k=100).

Copyright © 2026, All rights reserved.

EfficientVLA

Paper

❖ **EfficientVLA: Training-Free Acceleration and Compression for Vision-Language-Action Models (2026, NeruIPS)**

- Language module layer & visual token pruning / DiT step caching 을 통해 효율성 개선



EfficientVLA: Training-Free Acceleration and Compression for Vision-Language-Action Models

Yantai Yang[1,2] Yuhao Wang[1,3] Zichen Wen[1] Luo Zhongwei[1] Chang Zou[1,4] Zhipeng Zhang[1,5]
Chuan Wen[1,*] Linfeng Zhang[1,*]
[1]School of Artificial Intelligence, Shanghai Jiao Tong University
[2]Harbin Institute of Technology
[3]Xi'an Jiaotong University
[4]University of Electronic Science and Technology of China
[5]Anyverse Dynamics
Code: https://github.com/YantaiYang-05/EfficientVLA

Abstract

Vision-Language-Action (VLA) models, particularly diffusion-based architectures, demonstrate transformative potential for embodied intelligence but are severely hampered by high computational and memory demands stemming from extensive inherent and inference-time redundancies. While existing acceleration efforts often target isolated inefficiencies, such piecemeal solutions typically fail to holistically address the varied computational and memory bottlenecks across the entire VLA pipeline, thereby limiting practical deployability. We introduce **EfficientVLA**, a structured and training-free inference acceleration framework that systematically eliminates these barriers by cohesively exploiting multifaceted redundancies. EfficientVLA synergistically integrates three targeted strategies: (1) pruning of functionally inconsequential layers from the language module, guided by an analysis of inter-layer redundancies; (2) optimizing the visual processing pathway through a task-aware strategy that selects a compact, diverse set of visual tokens, balancing task-criticality with informational coverage; and (3) alleviating temporal computational redundancy within the iterative diffusion-based action head by strategically caching and reusing key intermediate features. We apply our method to a standard VLA model CogACT, yielding a 1.93× inference speedup and reduces FLOPs to 28.9%, with only a 0.6% success rate drop in the SIMPLER benchmark.

1 Introduction

Building upon advances in multimodal understanding from models integrating vision and language [1, 2, 3, 4, 5], Vision-Language-Action (VLA) models enable transformative embodied intelligence. These systems, such as OpenVLA [6], CogACT [7], π_0 [8] and RT-2 [?], directly translate multimodal inputs into executable actions, successfully tackling complex robotic manipulation and reasoning tasks using large-scale datasets [9, 10]. Many cutting-edge VLAs couple a Vision-Language Model (VLM) for scene and instruction parsing with a diffusion model to handle multi-modal action distribution [7, 11, 12, 13]. However, the significant computational and memory overheads of these Diffusion-based VLA architectures during the inference time pose critical barriers to their practical deployment, particularly for real-time interaction on resource-constrained robotic platforms.

Diffusion-based VLA architectures typically comprise a vision encoder to extract features, a large language model (LLM) [14, 15, 16, 17] core for multimodal reasoning, and a diffusion-based action

*Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Yang, Y., Wang, Y., Wen, Z., Zhongwei, L., Zou, C., Zhang, Z., ... & Zhang, L. (2026). Efficientvla: Training-free acceleration and compression for vision-language-action models. *Advances in Neural Information Processing Systems*, *38*, 40891-40914.

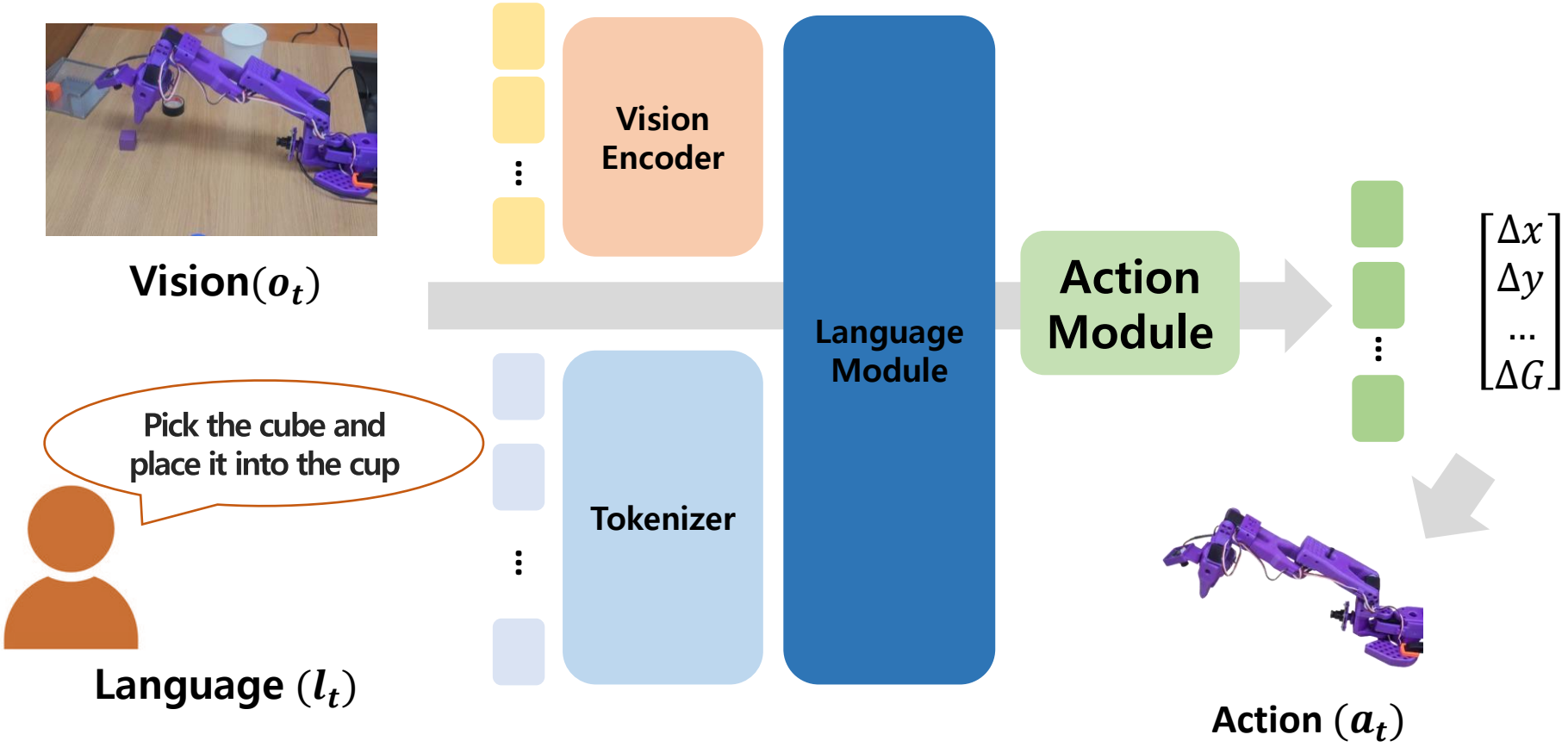
Copyright © 2026, All rights reserved.

DMQA

EfficientVLA

Introduction

❖ 기존 VLA efficiency 연구들은 어떤 한계를 가지고 있을까?

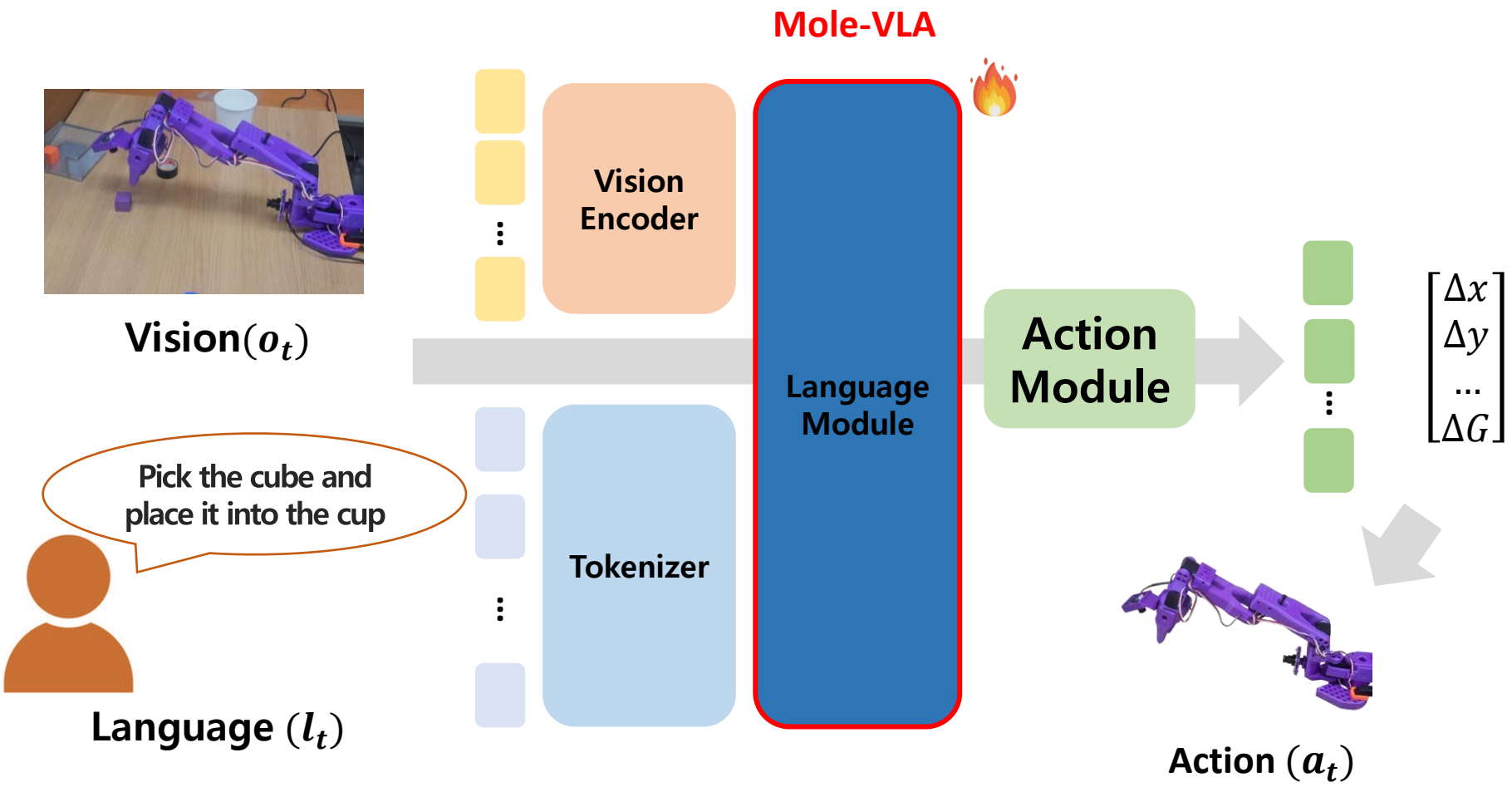


Copyright © 2026, All rights reserved.

EfficientVLA

Introduction

❖ 기존 VLA efficiency 연구들은 어떤 한계를 가지고 있을까?

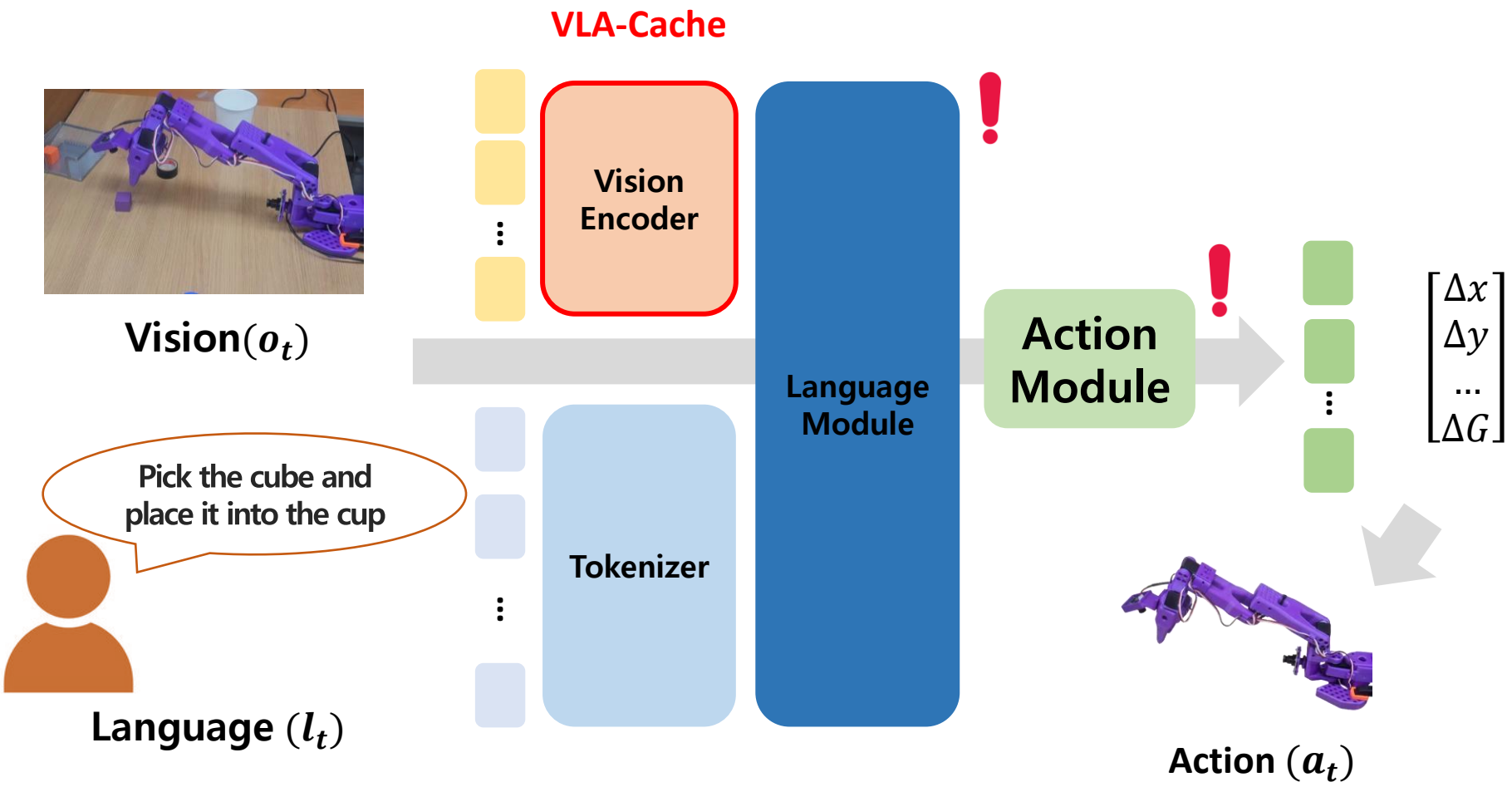


Copyright © 2026, All rights reserved.

EfficientVLA

Introduction

❖ 기존 VLA efficiency 연구들은 어떤 한계를 가지고 있을까?



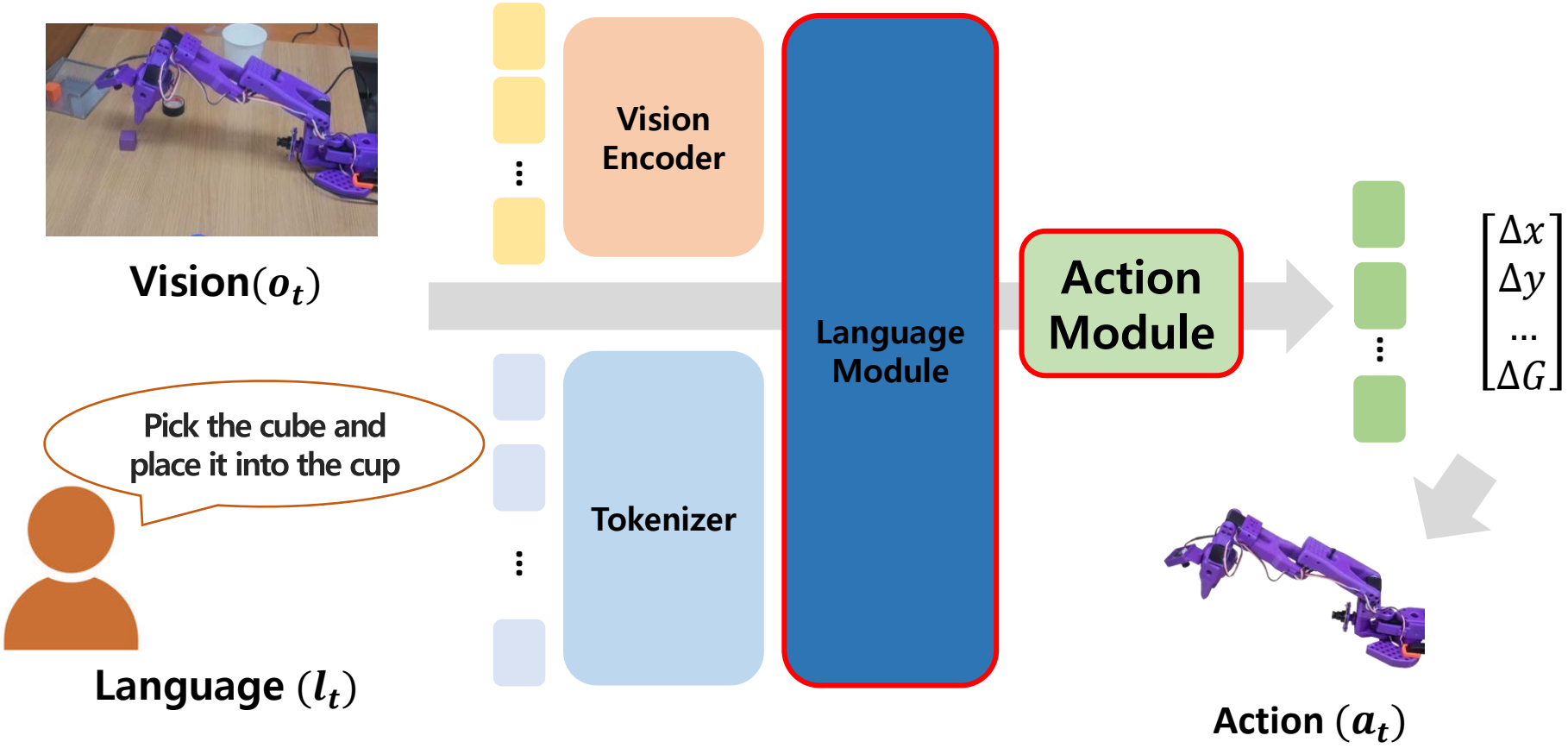
Copyright © 2026, All rights reserved.

DMQA

EfficientVLA

Introduction

❖ 어떻게 기존 VLA efficiency 연구들의 한계를 극복하고자 했을까?



Copyright © 2026, All rights reserved.

DMQA

EfficientVLA

Introduction

❖ **왜 Language와 action module에 초점을 맞췄을까?**

- Language module은 비교적 방대한 양의 파라미터 수를 가짐
- Diffusion 기반의 action module의 denoising step만 줄여도 추론 시간을 대폭 감소시킬 수 있음

	Vision Module	Language Module	Action Module
#Param (M)	802.3	6738.9	89.0
Vision Token	256	256	-
Denoising Steps	-	-	10
Inference Time (ms)	24.9	134.5	51.5
FLOPs (G)	405.50	3726.55	57.96

[CogACT (Baseline)]

	Vision Module	**Language Module**	**Action Module**
#Param (M)	802.3	**3971.1** (↓41%)	89.0
Vision Token	256	**56** (↓78%)	-
Denoising Steps	-	-	**2** (↓80%)
Inference Time (ms)	24.9	**58.9** (↓56%)	**26.2** (↓49%)
FLOPs (G)	405.50	**792.58** (↓78%)	**11.72** (↓80%)

[EfficientVLA]

Copyright © 2026, All rights reserved.

EfficientVLA

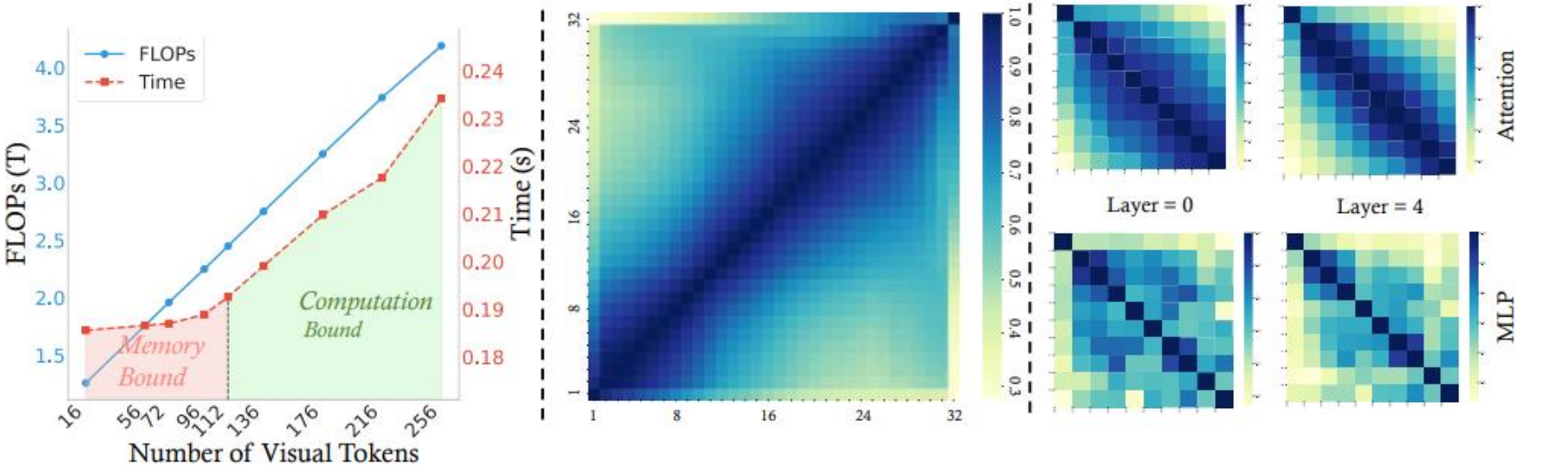
Introduction

❖ **왜 Language와 action module에 초점을 맞췄을까?**

a. Visual token pruning 만으로는 특정 구간 이후 추론 시간을 감소시키기 어려움

b. Language module 내에는 layer 간의 유사도가 매우 높은 layer들이 존재

c. Denoising step 마다의 output 간의 유사도가 매우 높은 step들이 존재



(a) Token-wise inference bottlenecks (b) Layer-wise output cosine similarity (c) Timestep-wise output cosine similarity

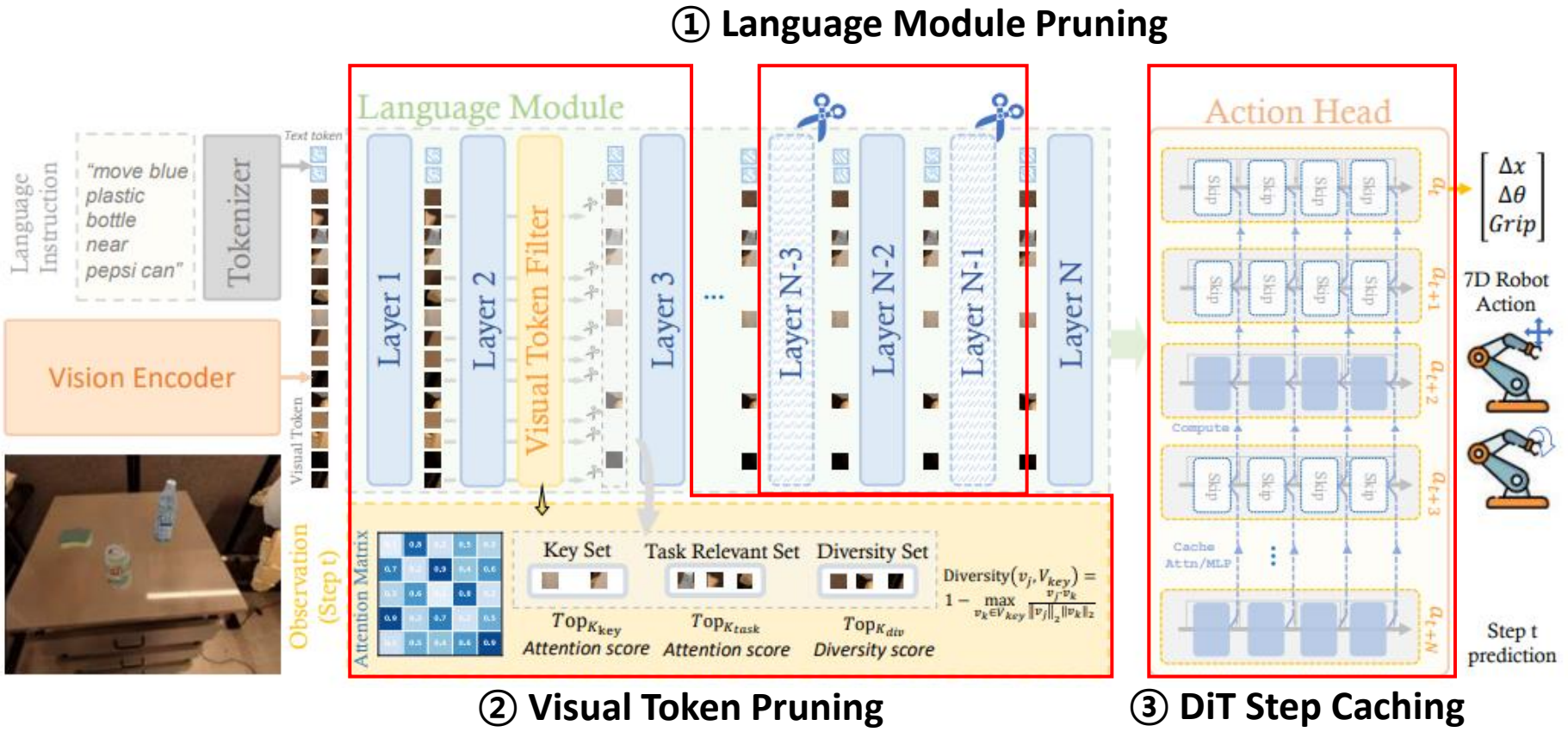
Copyright © 2026, All rights reserved.

DMQA

EfficientVLA

Overall Framework

❖ 어떻게 VLA efficiency 문제를 해결하였을까?



Copyright © 2026, All rights reserved.

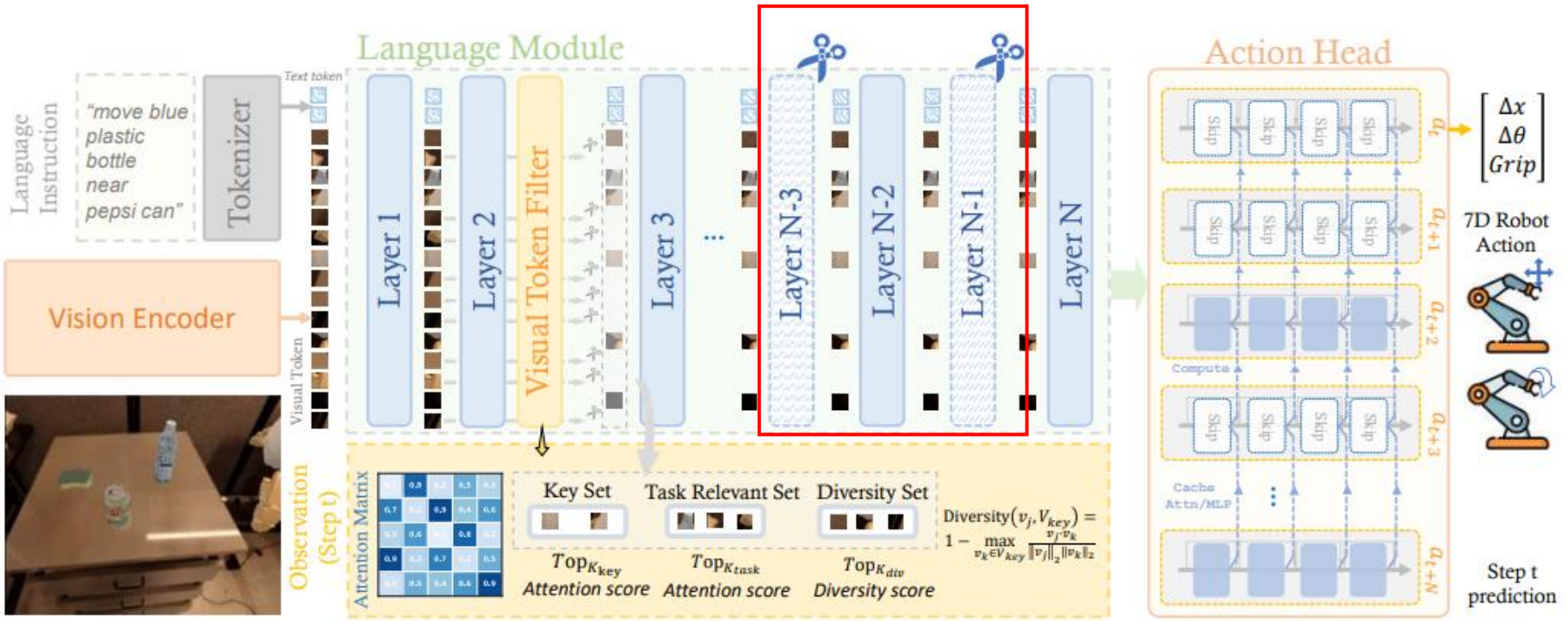
DMQA

EfficientVLA

Method

❖ **어떻게 language module pruning을 적용하였을까?**

- Layer 마다 입·출력값 사이의 유사도를 기반으로 중요도가 낮은 하위 n개 layer 제거



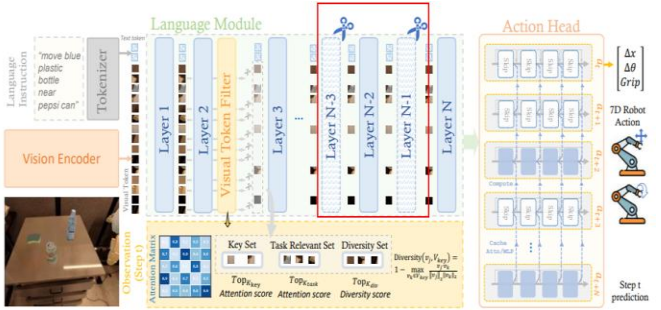
Copyright © 2026, All rights reserved.

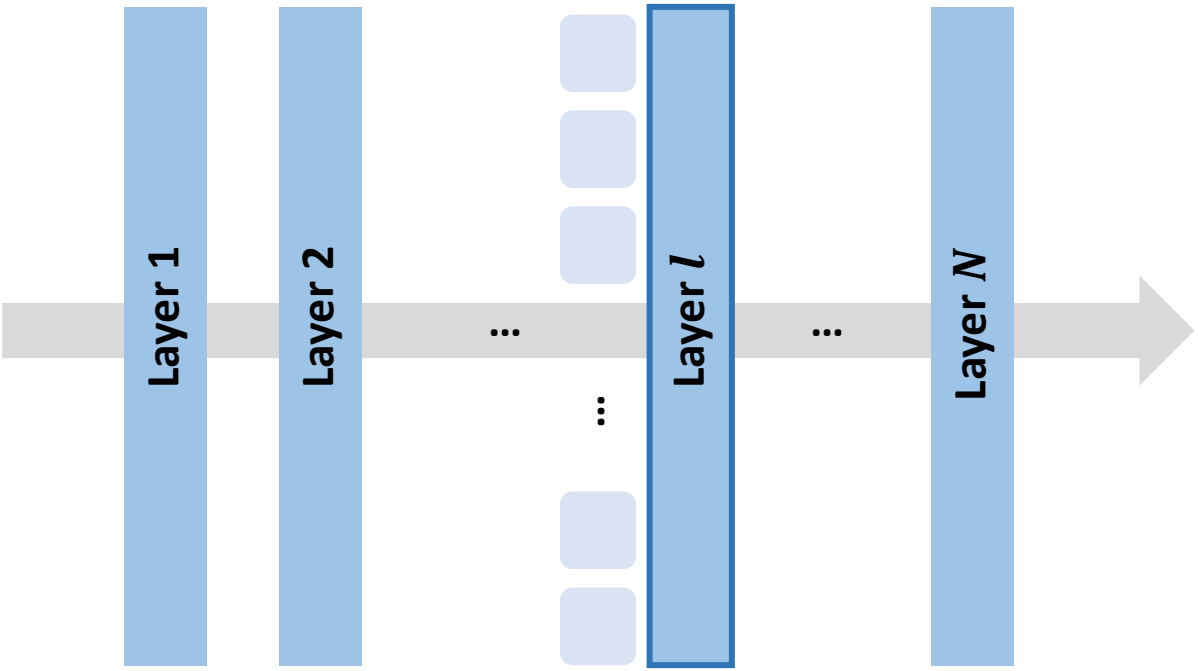
DMQA

EfficientVLA

Method

❖ 어떻게 language module pruning을 적용하였을까?





Copyright © 2026, All rights reserved.

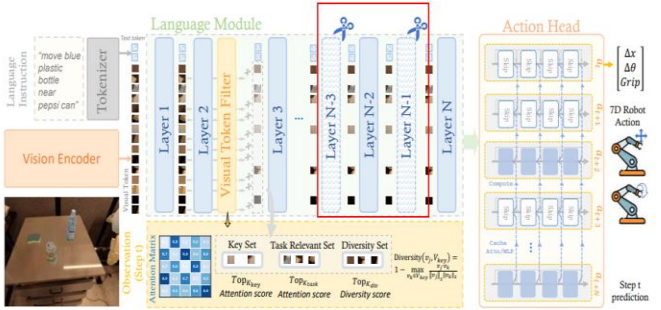
DMQA

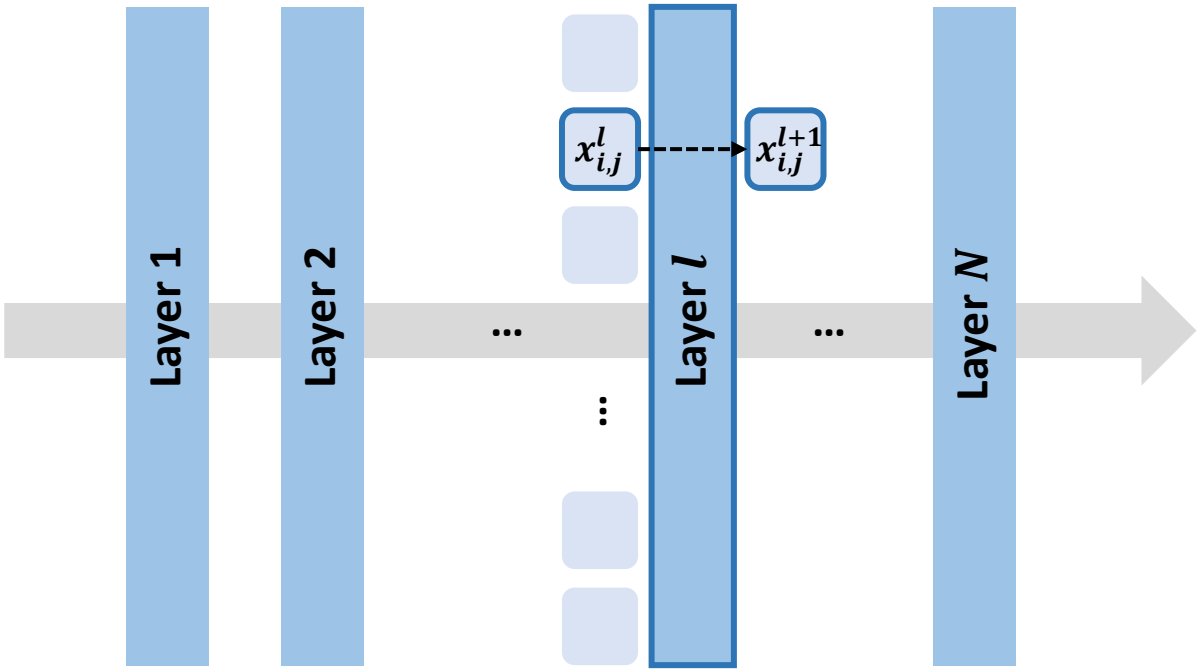
EfficientVLA

Method

❖ **어떻게 language module pruning을 적용하였을까?**

- i: sample index / j: token position index

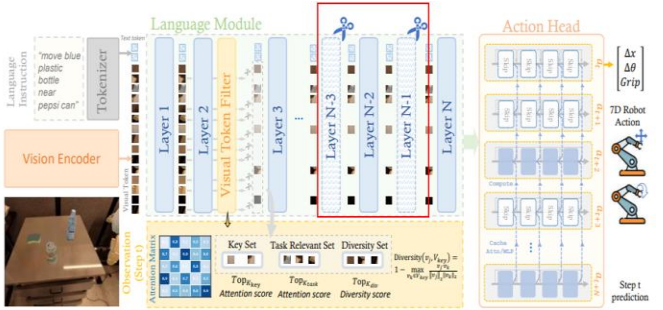




Copyright © 2026, All rights reserved.

DMQA

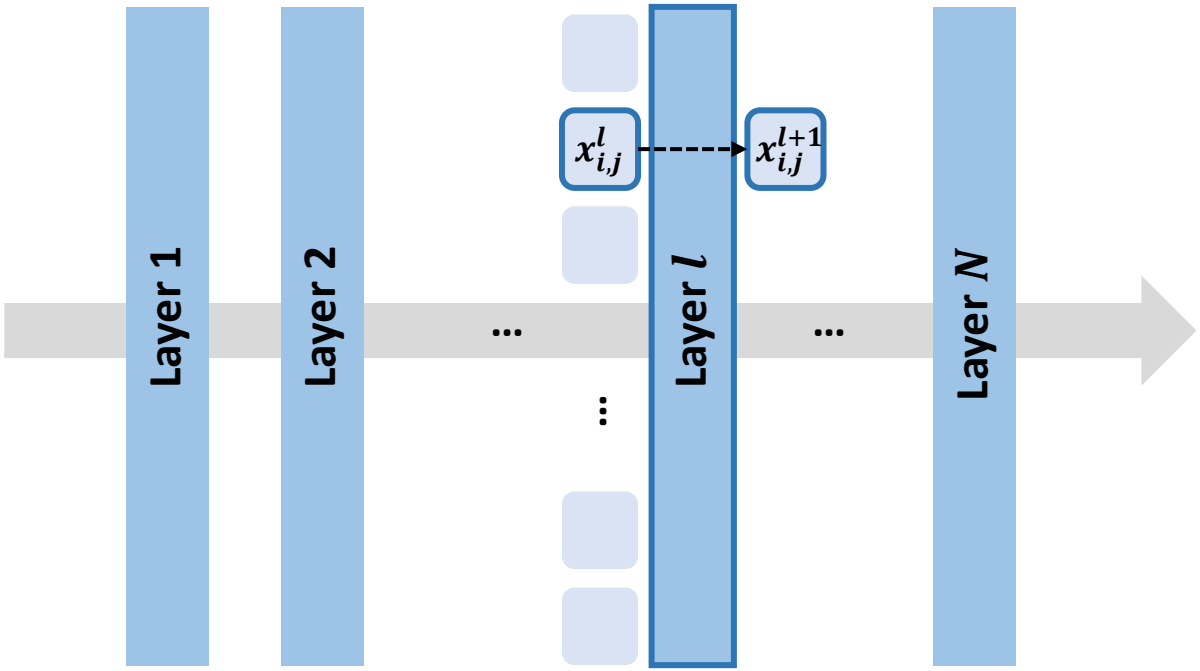
EfficientVLA



Method

❖ **어떻게 language module pruning을 적용하였을까?**

- i: sample index / j: token position index
- 입출력 값 사이의 유사도가 작을수록 importance score (I^l)는 커짐



유사도

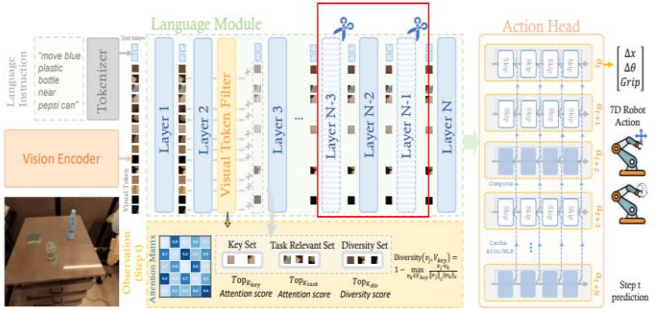
$$I^l = 1 - \frac{1}{|D|}\sum_{i=1}^{|D|}\left(\frac{1}{L}\sum_{j=1}^{L}\frac{x_{i,j}^{l}\,x_{i,j}^{l+1}}{\left\|x_{i,j}^{l}\right\|_2\left\|x_{i,j}^{l+1}\right\|_2}\right)$$

Copyright © 2026, All rights reserved.

DMQA

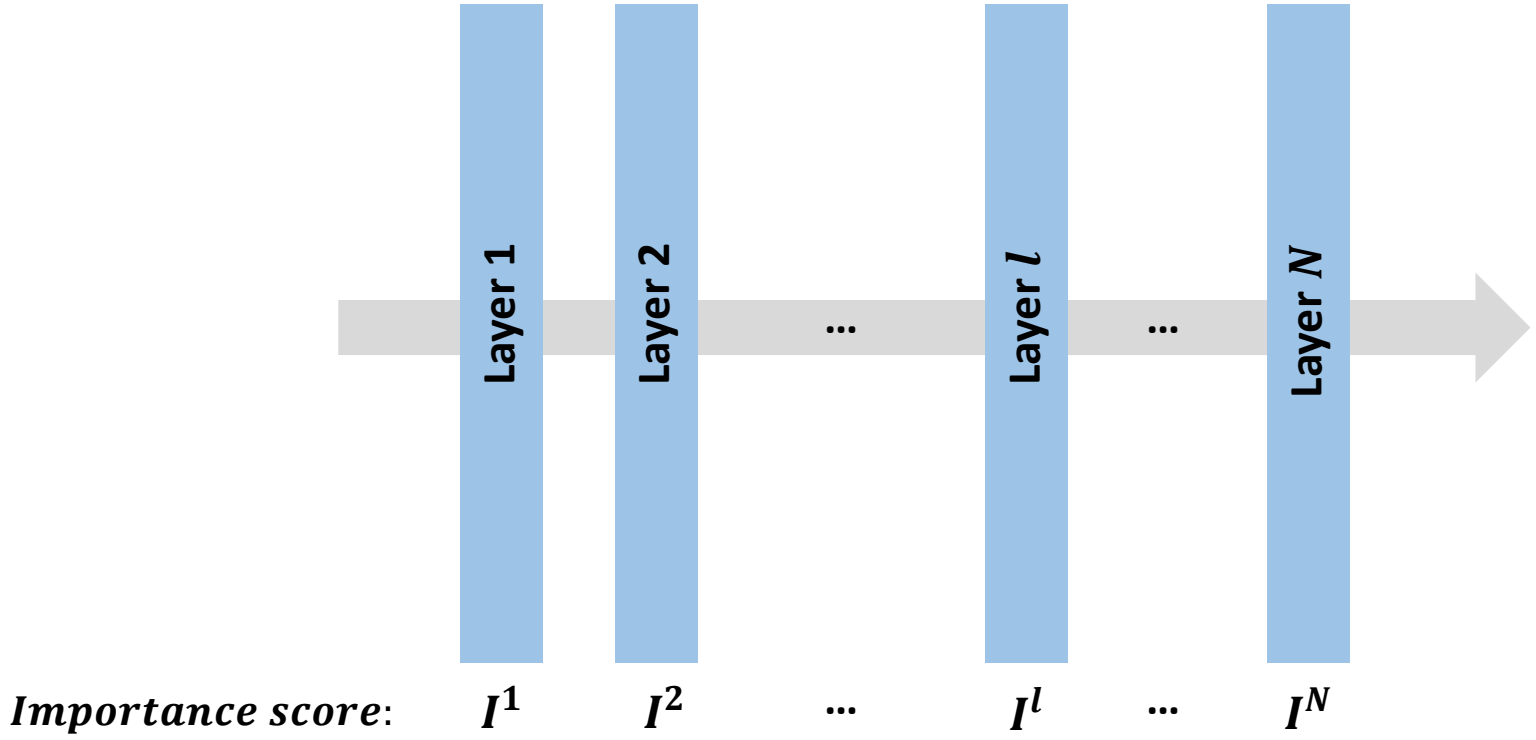
EfficientVLA

Method



❖ 어떻게 language module pruning을 적용하였을까?

- i: sample index / j: token position index
- 입출력 값 사이의 유사도가 작을수록 importance score (I^l)는 커짐

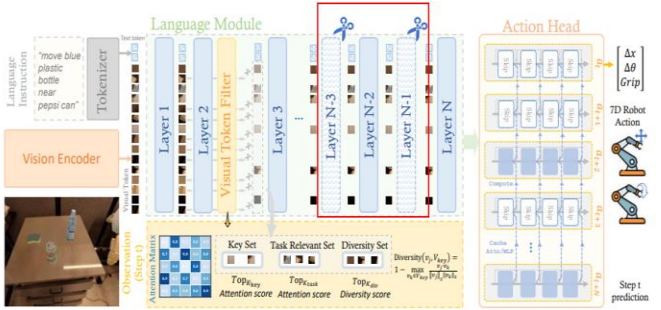


Copyright © 2026, All rights reserved.

DMQA

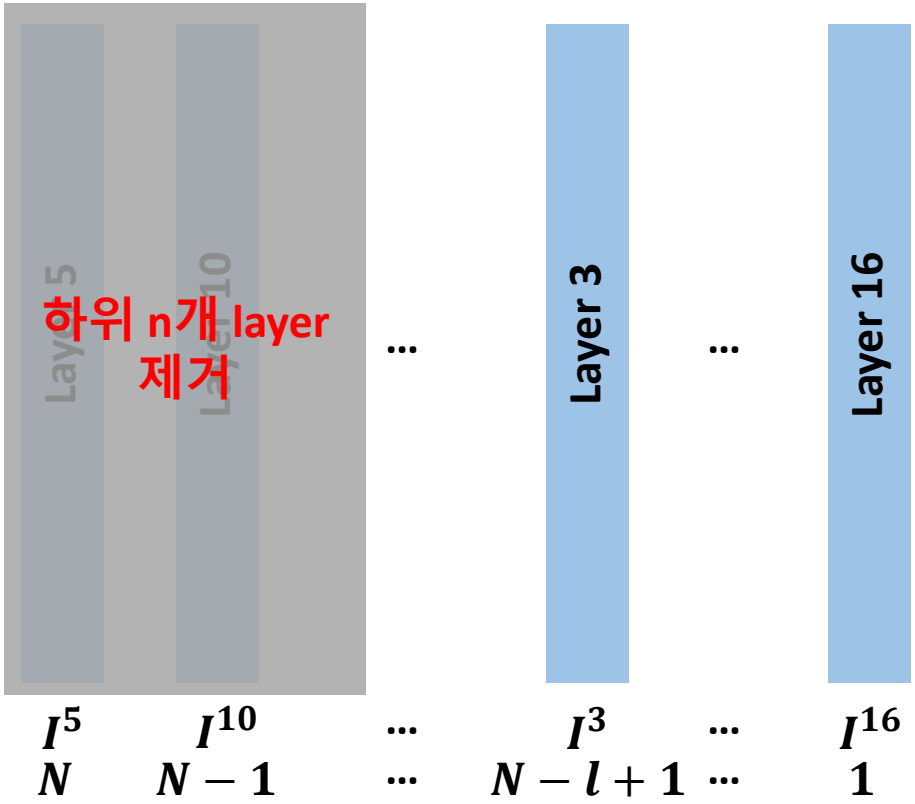
EfficientVLA

Method



❖ **어떻게 language module pruning을 적용하였을까?**

- i: sample index / j: token position index
- 입출력 값 사이의 유사도가 작을수록 importance score (I^l)는 커짐



$$\boldsymbol{Importance\ score}: \quad \boldsymbol{I^5} \quad \boldsymbol{I^{10}} \quad \boldsymbol{\cdots} \quad \boldsymbol{I^3} \quad \boldsymbol{\cdots} \quad \boldsymbol{I^{16}}$$

$$\boldsymbol{Rank}: \quad \boldsymbol{N} \quad \boldsymbol{N-1} \quad \boldsymbol{\cdots} \quad \boldsymbol{N-l+1} \quad \boldsymbol{\cdots} \quad \boldsymbol{1}$$

Copyright © 2026, All rights reserved.

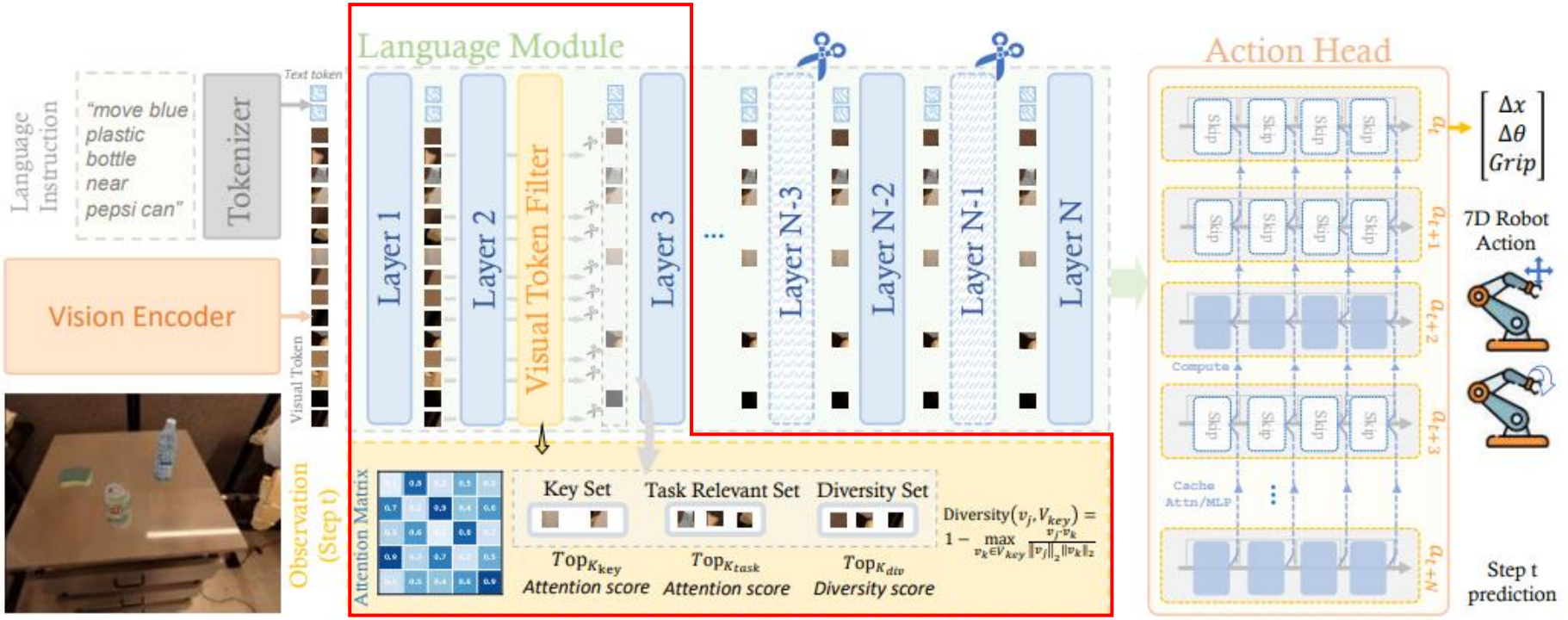
DMQA

EfficientVLA

Method

❖ **어떻게 visual token pruning을 적용하였을까?**

- VLA task와 관련성이 적은 token 제거
- 정보가 중복되는 token 제거

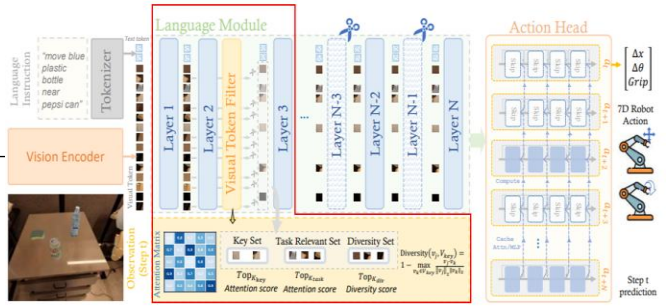


Copyright © 2026, All rights reserved.

DMQA

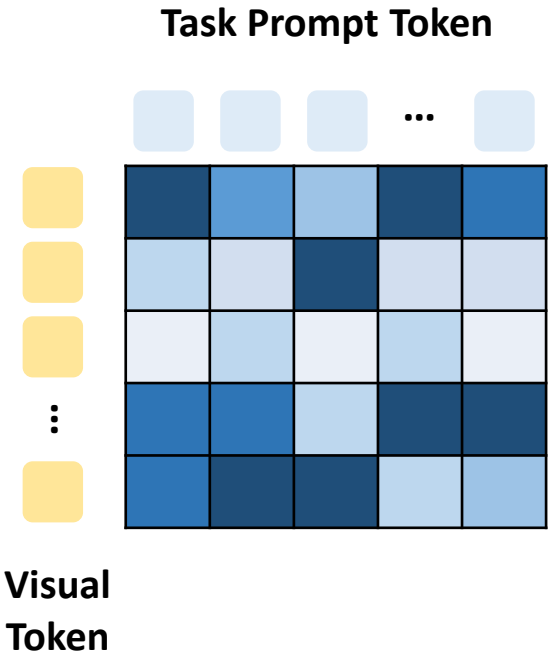
EfficientVLA

Method



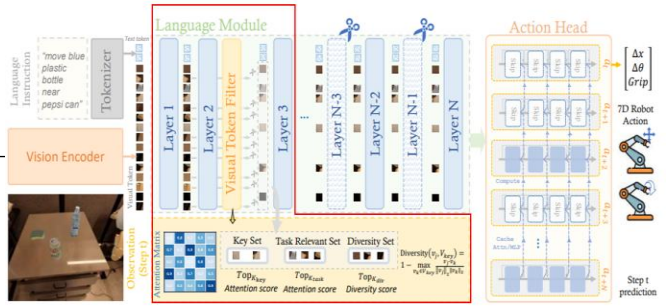
❖ **어떻게 visual token pruning을 적용하였을까?**

- Task prompt와의 **vision-to-text attention score**가 높은 상위 K_{key}개 token은 유지



Copyright © 2026, All rights reserved.

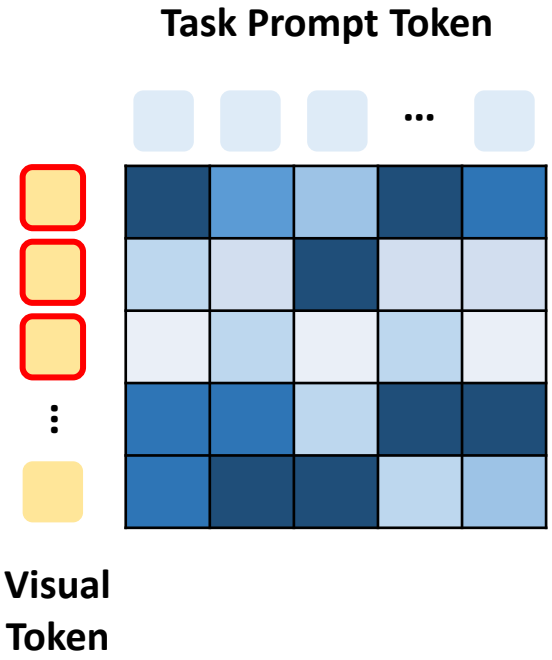
EfficientVLA



Method

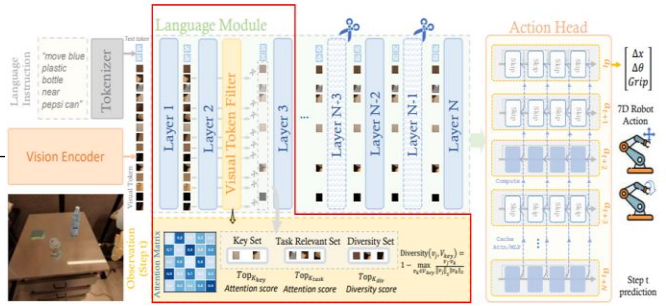
❖ **어떻게 visual token pruning을 적용하였을까?**

- Task prompt와의 attention score가 높은 상위 K_{key}개 token은 유지
  - ➢ 나머지 token들 중 attention score가 높은 상위 K_{task}개 token은 유지



Copyright © 2026, All rights reserved.

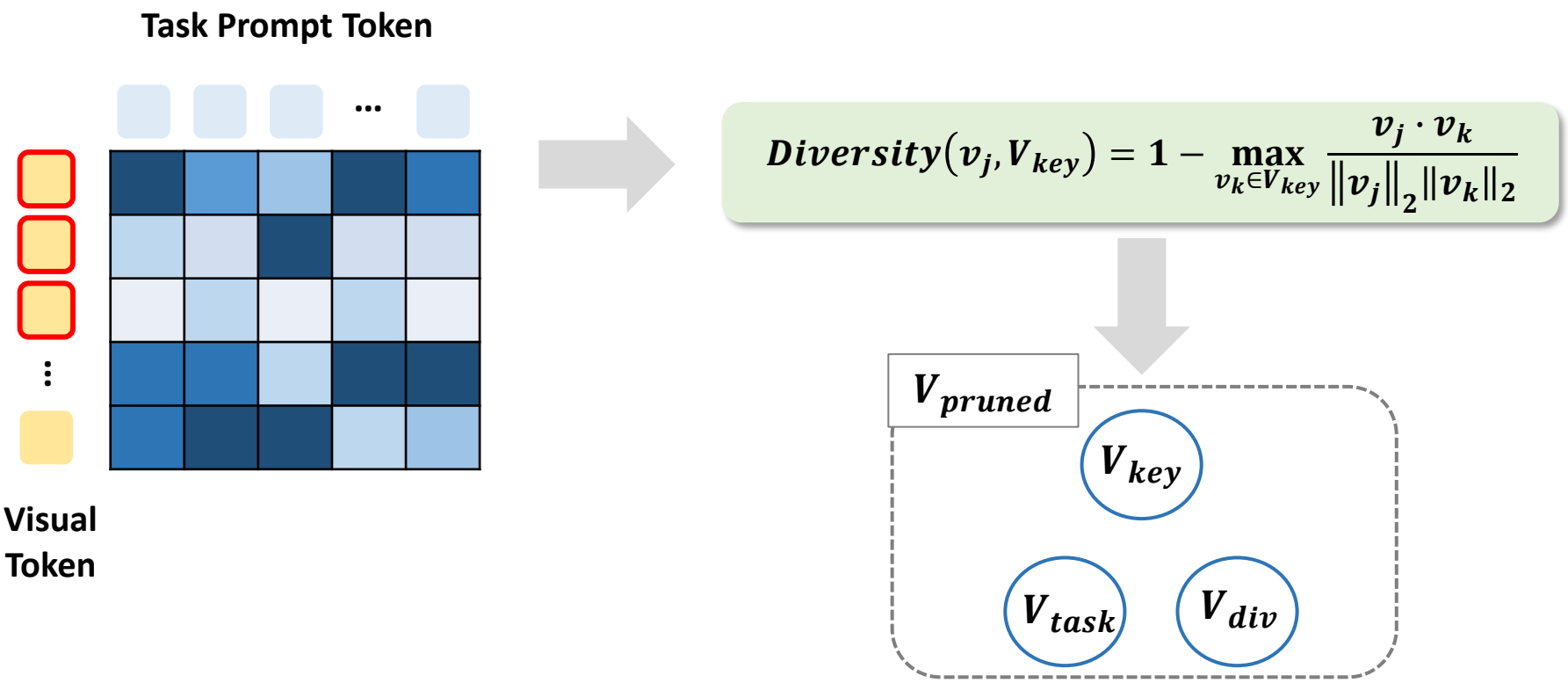
EfficientVLA



Method

❖ **어떻게 visual token pruning을 적용하였을까?**

- Task prompt와의 attention score가 높은 상위 K_{key}개 token은 유지
  - ➢ 나머지 token들 중 attention score가 높은 상위 K_{task}개 token은 유지
- 나머지 token들 중 key token들과의 유사도가 낮은(정보 중복 X) K_{div}개 token은 유지



Copyright © 2026, All rights reserved.

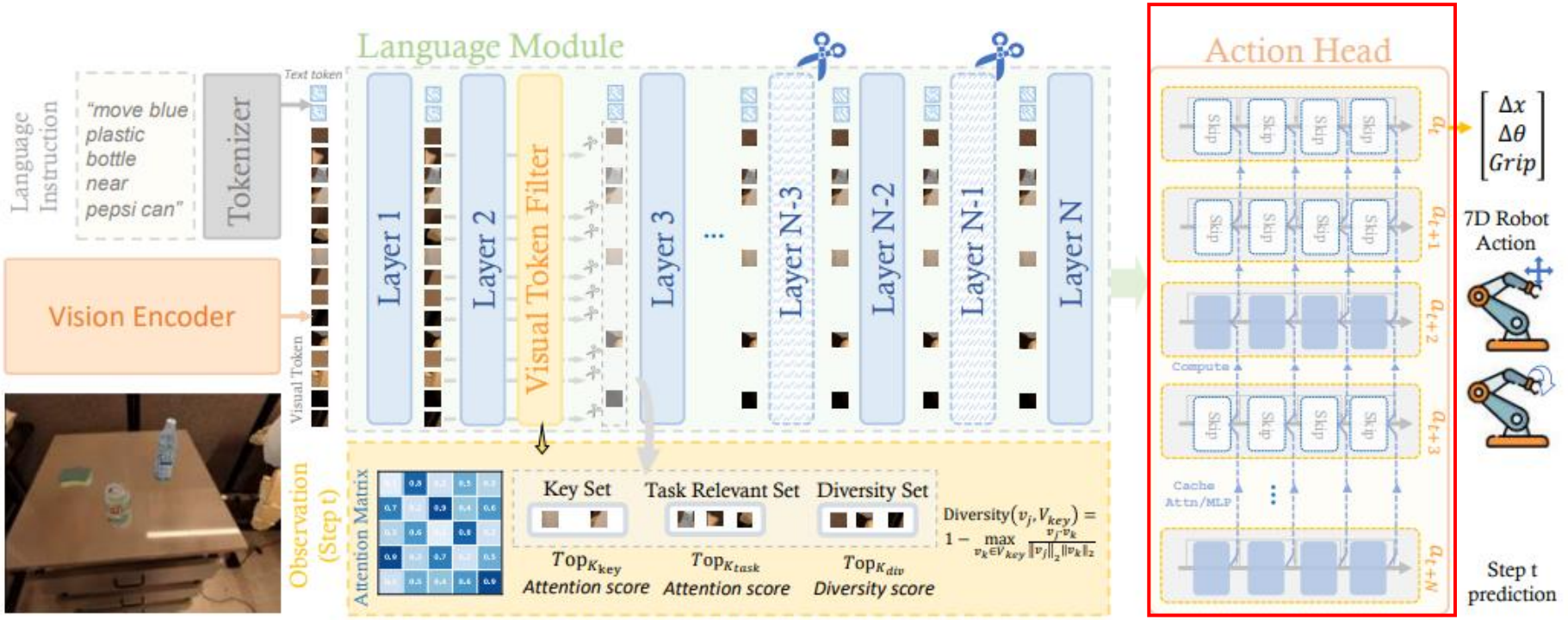
DMQA

EfficientVLA

Method

❖ **어떻게 DiT step caching을 적용하였을까?**

- 매 step마다가 아닌 특정 주기 (N step)마다 denoising update (Self-Attention, MLP)

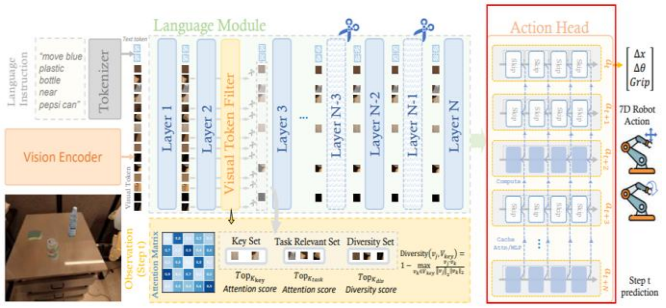


Copyright © 2026, All rights reserved.

DMQA

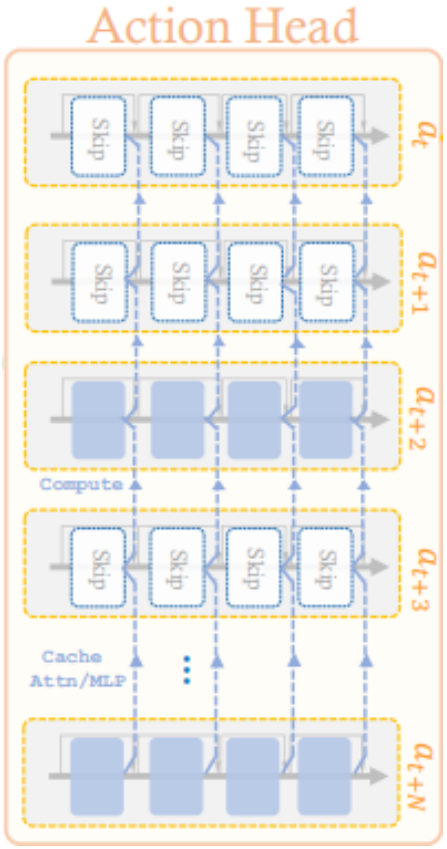
EfficientVLA

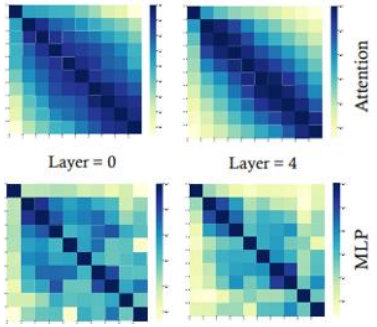
Method



❖ **어떻게 DiT step caching을 적용하였을까?**

- 매 step마다가 아닌 특정 주기 (N step)마다 denoising update (Self-Attention, MLP)





$$h_{t+N}^{attn} = SelfAttn(z_{t+N})$$

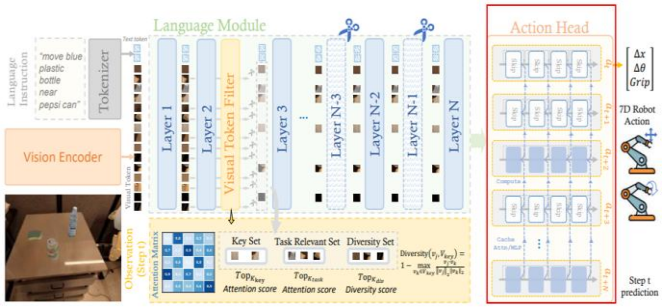
$$h_{t+N}^{mlp} = MLP(h_{t+N}^{attn} + z_{t+N})$$

Copyright © 2026, All rights reserved.

DMQA

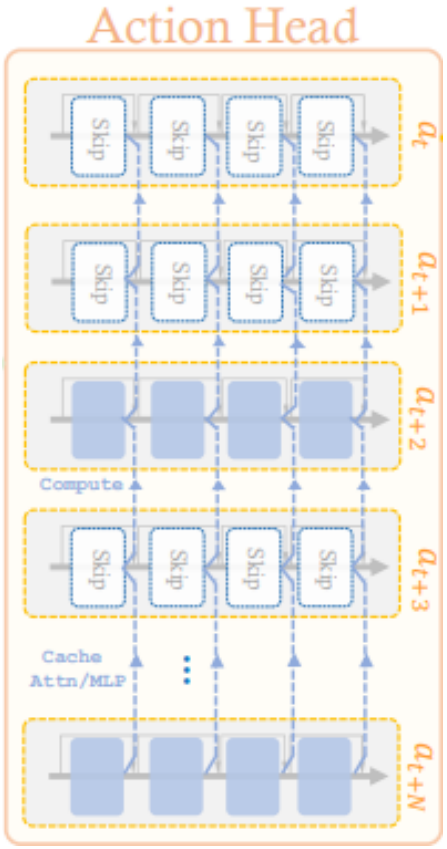
EfficientVLA

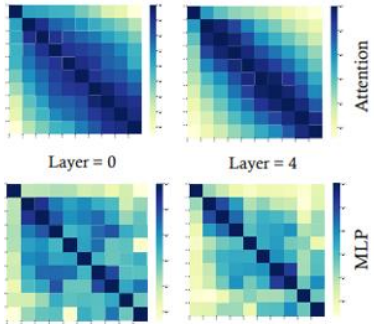
Method



❖ **어떻게 DiT step caching을 적용하였을까?**

- 매 step마다가 아닌 특정 주기 (N step)마다 denoising update (Self-Attention, MLP)





$$h_{t+s}^{attn} = h_{t+N}^{attn}$$

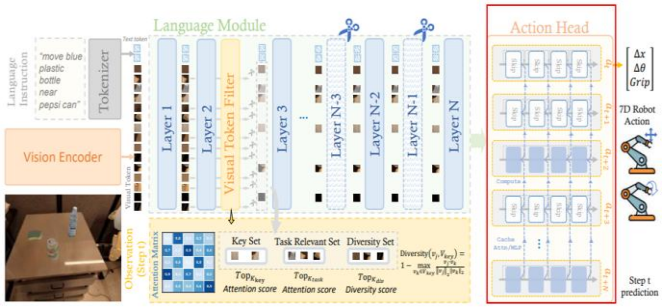
$$h_{t+s}^{mlp} = h_{t+N}^{mlp}$$

Copyright © 2026, All rights reserved.

DMQA

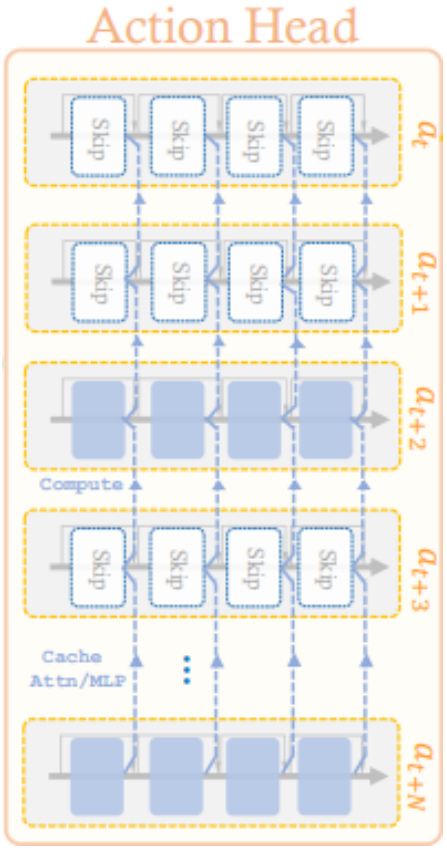
EfficientVLA

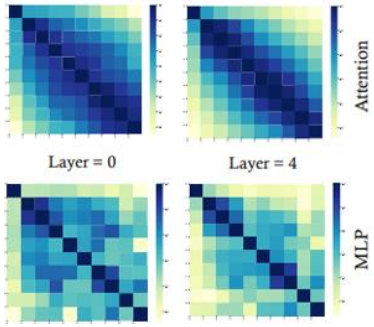
Method



❖ **어떻게 DiT step caching을 적용하였을까?**

- 매 step마다가 아닌 특정 주기 (N step)마다 denoising update (Self-Attention, MLP)





$$h_{t+2}^{attn} = SelfAttn(z_{t+2})$$

$$h_{t+2}^{mlp} = MLP(h_{t+2}^{attn} + z_{t+2})$$

Copyright © 2026, All rights reserved.

DMQA

EfficientVLA

Experiments

❖ **여러가지 환경에서 EfficientVLA의 성능은 어떨까?**

- Pruning을 많이 해도 성능은 최대한 유지하면서 효율성 개선

Table 2: Performance of EfficientVLA on the CogACT versus the other baselines in the SIMPLER environment. Settings vary by retained LLM layers (L) and visual tokens (T). *Random Dropping* denotes a method involving the random retention of 112 visual tokens.

SIMPLER	Method	Training-free	PickCan	MoveNear	Drawer	DrawerApple	Average	FLOPs↓	Speedup↑	Params (B)
Visual Matching	CogACT	-	91.3%	85.0%	71.8%	50.9%	74.8%	100.0%	1.00×	7.63
	Random Dropping	✓	9.7%	20.4%	53.5%	0.0%	20.9%	58.5%	1.20×	7.63
	FastV	✓	92.6%	81.4%	69.8%	52.4%	74.1%	42.0%	1.21×	7.63
	VLA-Cache	✓	92.0%	83.3%	70.5%	51.6%	74.4%	80.1%	1.38×	7.63
	EfficientVLA (L=28, T=112)	✓	95.3%	83.3%	70.3%	56.5%	76.4%	45.1%	1.59×	5.87
	EfficientVLA (L=28, T=56)	✓	94.7%	82.4%	69.8%	55.4%	75.5%	32.9%	1.71×	5.87
	EfficientVLA (L=22, T=112)	✓	94.0%	82.1%	69.2%	54.6%	75.0%	38.2%	1.78×	4.86
	EfficientVLA (L=22, T=56)	✓	93.3%	81.3%	68.2%	53.8%	74.2%	28.9%	1.93×	4.86
Variant Aggregation	CogACT	-	89.6%	80.8%	28.3%	46.6%	61.3%	100.0%	1.00×	7.63
	Random Dropping	✓	4.0%	16.1%	15.6%	0.0%	8.9%	58.5%	1.20×	7.63
	FastV	✓	91.4%	78.6%	27.6%	50.6%	62.1%	42.0%	1.19×	7.63
	VLA-Cache	✓	91.7%	79.3%	32.5%	45.8%	62.3%	82.6%	1.37×	7.63
	EfficientVLA(L=28, T=112)	✓	94.8%	77.6%	28.4%	51.9%	63.2%	45.1%	1.57×	5.87
	EfficientVLA (L=28, T=56)	✓	94.4%	77.2%	27.6%	51.3%	62.6%	32.9%	1.69×	5.87
	EfficientVLA (L=22, T=112)	✓	93.9%	76.4%	27.3%	50.6%	62.1%	38.2%	1.76×	4.86
	EfficientVLA (L=22, T=56)	✓	93.2%	75.8%	26.9%	49.2%	61.2%	28.9%	1.91×	4.86

Copyright © 2026, All rights reserved.

EfficientVLA

Experiments

❖ **Visual token ratio와 cache interval에 따라 EfficientVLA의 성능은 어떨까?**

- Pruning / cache interval을 증가시켜도 성능은 최대한 유지하면서 효율성 개선
- 많은 양의 정보를 다 활용한다고 성능이 항상 좋은 것이 아님을 확인

Table 4: Performance impact of varying visual token reduction ratios (Left) and diffusion action head cache intervals (Right) as applied within our EfficientVLA framework.

Token Ratio	56 77.8%	72 71.8%	96 62.5%	112 56.2%	256 100.0%
Accuracy	95.0%	95.3%	95.0%	96.0%	91.3%
Inference time (s)	0.1866	0.1870	0.1889	0.1956	0.2342
FLOPs (T)	1.76	1.96	2.25	2.45	4.19

Cache Interval	1	2	3	4	5
Accuracy	91.3%	94.0%	93.7%	90.3%	93.7%
Inference time (s)	0.2342	0.2031	0.1987	0.1953	0.1909
FLOPs (T)	4.190	4.161	4.155	4.150	4.144

Copyright © 2026, All rights reserved.

EfficientVLA

Experiments

❖ **EfficientVLA의 각 component 별 효과는 어떨까?**

- Visual token이 아닌 language module에 pruning을 적용하는 것이 효율적
- DiT step caching의 효과도 같이 확인할 수 있음

Table 5: **Ablation study** on our EfficientVLA, where 'Layer' denotes applying only the LLM layer pruning component, and 'MLP' refers to a distinct strategy of compressing 25% of MLP weights within each layer.

	Model Compression		Visual Token	Action	Success	Inference	Speedup↑
	Layer	MLP	Pruning	Cache	Rate	Time (s)	
Ex0	✗	✗	✗	✗	91.3%	0.2342	1.00×
Ex1	✗	✗	✓	✗	95.6%	0.1866	1.25×
Ex2	✗	✗	✗	✓	93.7%	0.1909	1.23×
Ex3	✓	✗	✓	✗	85.7%	0.1604	1.46×
Ex4	✓	✓	✗	✗	92.3%	0.1638	1.43×
Ex5	✓	✓	✗	✓	93.3%	0.1387	1.69×
Ex6	✗	✗	✓	✓	**95.3%**	0.1592	1.47×
Ex7	✓	✓	✓	✓	93.3%	**0.1213**	**1.93**×

Copyright © 2026, All rights reserved.

EfficientVLA

Experiments

❖ **EfficientVLA이 일반적인 시각-언어 이해에서도 효과적인가?**

- 중요한 token / layer들의 정보만 사용함에 따라 우수한 시각-언어 이해 성능을 보임

Table 6: Performance evaluation of the EfficientVLA token pruning strategy compared to the vanilla LLaVA-1.5-7B baseline and the traditional FastV method across diverse multimodal vision-language benchmarks.

Method	Benchmarks										Avg
	GQA	MMB	MMB-CN	MME	POPE	SQA	VQA-v2	VQA-Text	VizWiz	OCRBench	
LLaVA-1.5-7B (Vanilla, 576 tokens)	61.9	64.7	58.1	1862	85.9	69.5	78.5	58.2	50.0	297	100.0%
+FastV (192 tokens)	52.7	61.2	57.0	1612	64.8	67.3	67.1	52.5	50.8	291	91.2%
+Ours (192 tokens)	58.5	62.3	57.0	1823	82.8	69.6	76.7	57.6	51.0	292	98.0%
+FastV (128 tokens)	49.4	56.1	56.4	1490	56.0	60.2	61.8	50.6	51.3	285	86.4%
+Ours (128 tokens)	58.2	61.5	56.6	1763	79.8	68.5	75.7	56.1	51.4	289	96.6%

Copyright © 2026, All rights reserved.

Conclusion

Copyright © 2023, All rights reserved.

Conclusion

❖ Background

- VLM의 발전에 이어 VLA 모델들이 다양한 로봇 task에서 우수한 성능을 보이고 있음
- 추가적인 학습이 없으면서 효율적인 VLA 모델 구축을 위한 연구들이 이어 발전하고 있음

❖ VLA-Cache

- Patch (token)별 유사도와 cross-attention score를 기반으로 각 시점마다 크게 변하지 않는 token들은 재사용 함으로써 효율성을 보장

❖ EfficientVLA

- Language module과 diffusion 기반 action module의 경량화에 초점을 둠

❖ Conclusion

- 어렵지 않은 기법으로 성능은 최대한 유지하면서 효율성 측면을 개선함

Copyright ⓒ 2026, All rights reserved.

References

Copyright © 2023, All rights reserved.

References

- Zhang, D., Sun, J., Hu, C., Wu, X., Yuan, Z., Zhou, R., ... & Zhou, Q. (2025). Pure vision language action (vla) models: A comprehensive survey. arXiv preprint arXiv:2509.19012.

- Xu, S., Wang, Y., Xia, C., Zhu, D., Huang, T., & Xu, C. (2026). Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. Advances in Neural Information Processing Systems, 38, 164448-164473.

- Yang, Y., Wang, Y., Wen, Z., Zhongwei, L., Zou, C., Zhang, Z., ... & Zhang, L. (2026). Efficientvla: Training-free acceleration and compression for vision-language-action models. Advances in Neural Information Processing Systems, 38, 40891-40914.

Copyright © 2026, All rights reserved.

고맙습니다

Copyright © 2023, All rights reserved.