Next Generation VLA: From Reasoning to Efficiency

A Review of CogACT & EfficientVLA

2026 동계 세미나

Sogang University

Vision & Display Systems Lab., Dept. of Electronic Engineering

Presented By

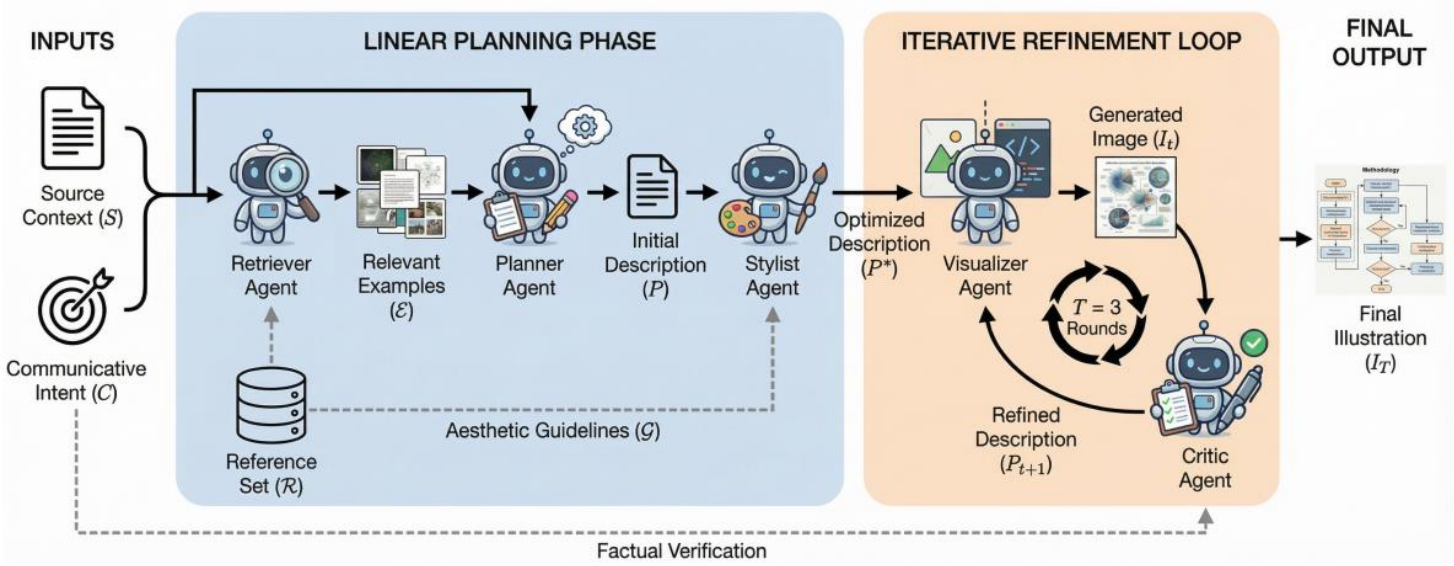
이종범

Outline

- PaperBanana: Automating Academic Illustration for AI Scientists (Arxiv 2026)
- Background
  - VLA 모델의 진화 과정 (Gen 1 vs Gen 2)
  - 기존 모델의 한계점: 인지적 간극 및 효율성 문제
- CogACT: A Foundational Vision-Language-Action Model for Synergizing Cognition and Action in Robotic Manipulation (Arxiv 2024)
- EfficientVLA: Training-Free Acceleration and Compression for Vision-Language-Action Models (NeurIPS 2025)
- Conclusion

PaperBanana

- Problem
  - 논문용 Publication-Ready Figure 는 여전히 수작업 병목 발생
  - Code-based는 표현력 한계, Image Generation은 학술 기준 일관성·정확성 문제
- Solution: 5개의 Agent로 구성
  - Linear Planning Phase: 레퍼런스 검색 → 계획 → 스타일 최적화
  - Iterative Refinement Loop: Visualizer ↔ Critic (3 rounds)



< PaperBanana Framework >

PaperBanana

- Experiment

Agent	실제 호출 모델	역할
Retriever	Gemini-3-Pro (VLM)	Reference 검색
Planner	Gemini-3-Pro (VLM)	Description 작성
Stylist	Gemini-3-Pro (VLM)	Style 최적화
Visualizer	Nano-Banana-Pro / GPT-Image-1.5	이미지 생성
Critic	Gemini-3-Pro (VLM)	생성 이미지 평가

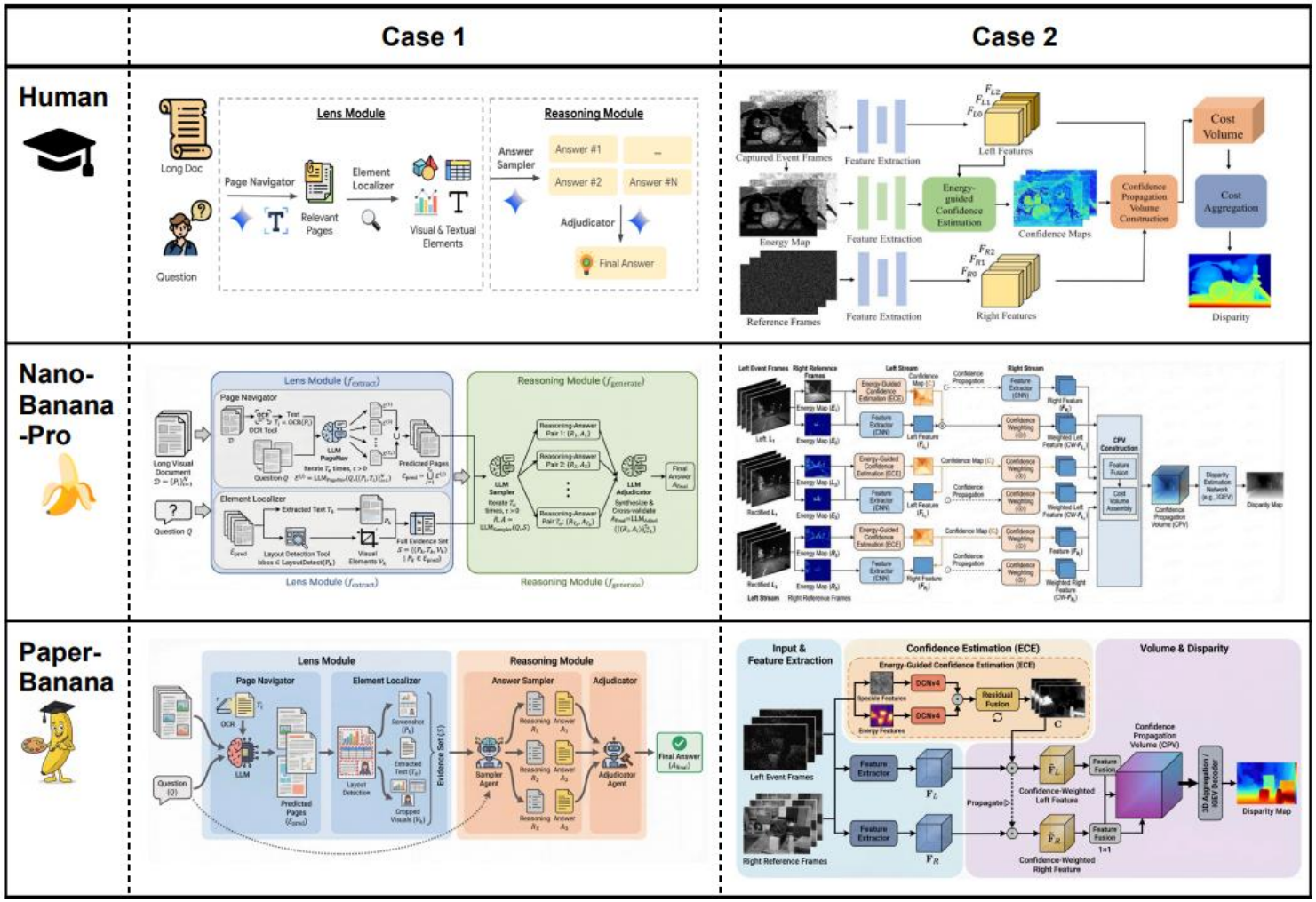
< PaperBanana Experiment Setting >

Method	Faithfulness ↑	Conciseness ↑	Readability ↑	Aesthetic ↑	Overall ↑
Vanilla Settings					
GPT-Image-1.5	4.5	37.5	30.0	37.0	11.5
Nano-Banana-Pro	43.0	43.5	38.5	65.5	43.2
Few-shot Nano-Banana-Pro	41.6	49.6	37.6	60.5	41.8
Agentic Frameworks					
Paper2Any (w/ Nano-Banana-Pro)	6.5	44.0	20.5	40.0	8.5
PAPERBANANA (Ours)					
w/ GPT-Image-1.5	16.0	65.0	33.0	56.0	19.0
w/ Nano-Banana-Pro	45.8	**80.7**	**51.4**	**72.1**	**60.2**
Human	**50.0**	50.0	50.0	50.0	50.0

< PaperBanana 성능 비교표 >

PaperBanana

- Qualitative Results



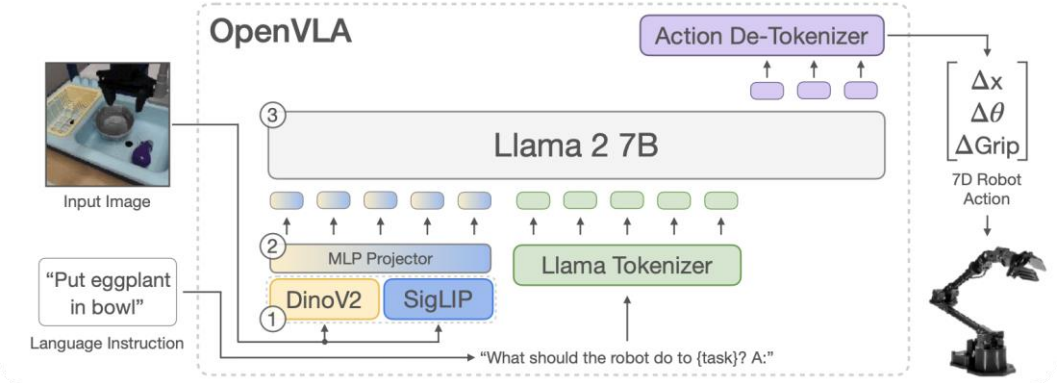
< PaperBanana로 생성된 그림 >

Background

- What is VLA?
  - Vision-Language-Action (VLA) Models
    - 대규모 언어 모델(LLM)과 시각 인코더(Vision Encoder)를 결합한 VLM을 로봇 제어 영역(Action)으로 확장한 파운데이션 모델
    - LLM의 추론 능력과 시각적 이해를 물리적 행동으로 변환
  - Input & Output Mechanism
    - Input: RGB Image + Text Prompt (“Pick up the apple”)
    - Output: Robot Action Token 7-DoF ($Pos_x, Pos_y, Pos_z, Rot_x, Rot_y, Rot_z, Gripper$)
  - Significance
    - 일반화: 대규모 데이터 학습을 통해, Zero-shot 환경에서도 대응 능력 확보
    - 기존 Rule-based 방식의 한계를 넘어선 범용 로봇 제어 실현

Background

- Vision-Language-Action Model (Gen 1)
  - Pradigm: “Big Data + Transformer”
    - Data Scaling: 인터넷 규모의 데이터 학습을 통해 로봇의 일반화 능력 확보
    - Representative Models: RT-2, Octo, OpenVLA
  - Open X-Embodiment Dataset
    - 다양한 형태와 환경 데이터를 하나의 거대 모델에 통합
  - Mechanism: “Next-Token Prediction”
    - Vision & Text → Tokenization → Transformer
    - LLM이 자기 회귀 방식으로 단어를 예측하듯, 로봇의 행동을 Token단위로 예측



< OpenVLA Framework >

Background

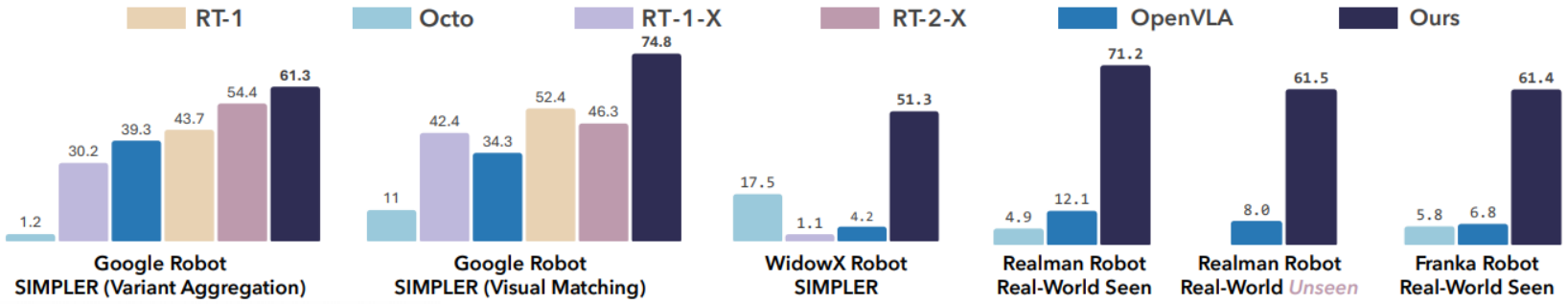
- Limitations of Gen 1 (Problem Definition)
  - Problem 1: The Cognitive Gap (Lack of Reasoning)
    - Direct Mapping Issue: Text + Vision에서 low-level Action으로 바로 연결되는 구조
      - Input에 대해 계획 없이 즉각적으로 다음 Action만 생성하므로, 복잡하거나 긴 순서가 필요한 작업에 매우 취약
    - Missing "CoT": 복잡한 작업 수행 시, "Why"에 대한 중간 추론 과정 부재
      - LLM의 추론 능력을 버리고 Action만 예측하므로, 낯선 환경에서 실패율 급증
  - Problem 2: Deployment Inefficiency (Latency & Cost)
    - High Computational Cost: 성능 확보를 위해 7B의 거대 파라미터로 인한 메모리 부담
    - Low Inference Speed: 추론 속도가 너무 느려 (5Hz ↓) 실시간 로봇 제어에 부적합

1) Li, Qixiu, et al. "CogACT: A Foundational Vision-Language-Action Model for Synergizing Cognition and Action in Robotic Manipulation." arXiv preprint arXiv:2411.19650 (2024).

CogACT: A Foundational Vision-Language-Action Model for Synergizing Cognition and Action in Robotic Manipulation [Arxiv 2024]

CogACT

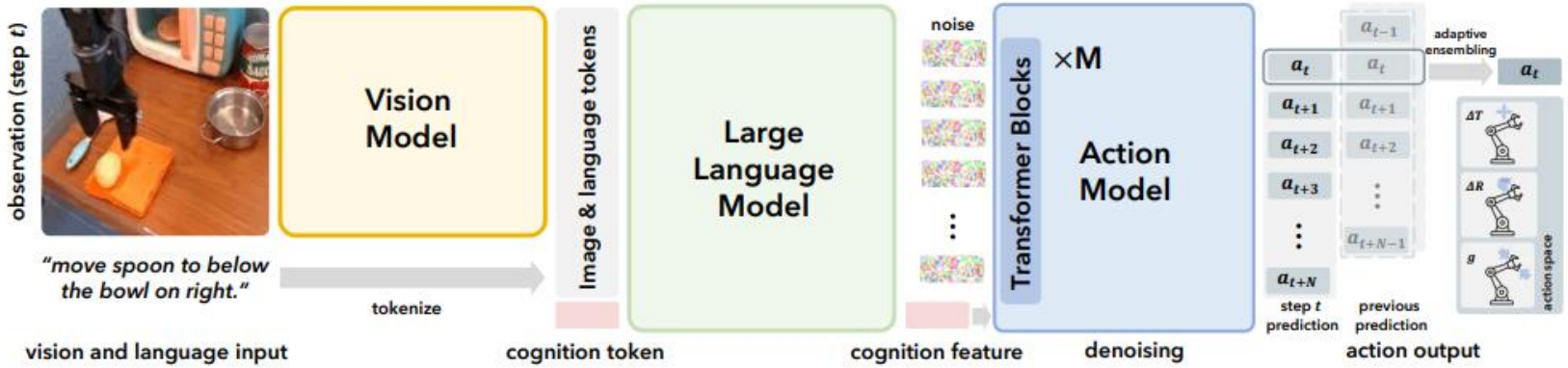
- Introduction
  - VLA Gen 1 State
    - 기존 모델(RT-2, OpenVLA)은 Text + Vision에서 low-level Action으로 바로 연결 구조
    - $P(a_t|I_t, inst)$
  - Problem: Cognitive Gap
    - 복잡한 작업에서는 '현재 상태 분석'과 '계획 수립'이 선행되어야 함
    - 중간 추론 단계 부재로 인해 낯선 환경이나 긴 작업에서 실패율 급증
  - Goal
    - 명시적인 Cognition(인식/추론) 단계를 도입하여 행동의 정확도 향상



< Direct Mapping vs Explicit >

CogACT

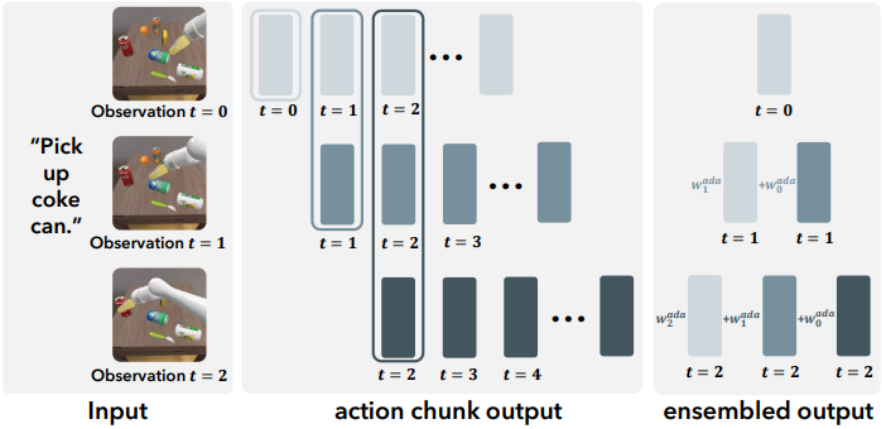
- Overall Architecture
  - Vision Model: 시각 정보 추출 (DINOv2 + SigLIP)
    - DINOv2: 이미지 자체의 구조적 공간적 특징을 매우 정교하게 포착하는 담당
    - SigLIP: 이미지와 언어 명령 사이의 의미적 연결을 담당
  - Large Language Model: LLaMA-2-7B
    - Language model은 Text, Visual, Cognition Token을 concat하여 입력받음
    - Output으로 '현재 관측 + 지시문을 통합한 인지 표현'인 Cognition feature를 Action Model의 condition으로 전달



< CogACT Framework >

CogACT

- Overall Architecture
  - Diffusion Action Module(DiT)
    - Vision & Language model의 output인 f_t^c를 Action Model의 condition으로 입력
    - 로봇의 연속적이고 multi-modal인 행동 특성을 diffusion 과정으로 모델링
    - 현재 Action(a_t)뿐 아니라 향후 N개 step의 Action을 함께 예측 ($N = 15$, 논문 기본값)
  - Adaptive Action Ensemble (AAE)
    - 매 시점 t 관측뿐만 아니라 과거 K개 관측 시점의 예측도 결합
      - 현재: $a_t^{\{o_t\}}$
      - 과거: $a_t^{\{o_{\{t-1\}}\}}, a_t^{\{o_{\{t-2\}}\}}, \dots, a_t^{\{o_{\{t-K\}}\}}$
      - 최종: $\Sigma_{k=0}^{K} w_k^{ada} \cdot a_t^{o_{t-k}}$
      - $w_k^{ada} = \exp\left(\beta \cdot \cos\left(a_t^{o_t}, a_t^{o_{t-k}}\right)\right)$
        (β=0.1, 논문 기본값)
      - 현재 예측과 유사한 과거 예측에 큰 가중치



< AAE >

※ Condition: Action model이 Action을 만들 때 참고하는 Context 정보

CogACT

- Experiment – Simulation Benchmark
  - SIMPLER: 로봇 조작 정책을 시뮬레이션에서 공정하게 평가하기 위한 벤치마크
  - Google Robot
    - Visual Matching: 현실과 유사한 조건
    - Variant Aggregation: 배경/조명/객체 변화

Google Robot	Method	Pick Coke Can	Move Near	Open/Close Drawer	Open Top Drawer and Place Apple	*Average*
SIMPLER (Visual Matching)	RT-1 [7]	85.7	44.2	73.0	6.5	52.4
	RT-1-X [48]	56.7	31.7	59.7	21.3	42.4
	RT-2-X [48]	78.7	77.9	25.0	3.7	46.3
	Octo-Base [62]	17.0	4.2	22.7	0.0	11.0
	OpenVLA [30]	18.0	56.3	63.0	0.0	34.3
	Ours	**91.3**	**85.0**	**71.8**	**50.9**	**74.8**
SIMPLER (Variant Aggregation)	RT-1 [7]	89.8	50.0	32.3	2.6	43.7
	RT-1-X [48]	49.0	32.3	29.4	10.1	30.2
	RT-2-X [48]	82.3	79.2	**35.3**	20.6	54.4
	Octo-Base [62]	0.6	3.1	1.1	0.0	1.2
	OpenVLA [30]	60.8	67.7	28.8	0.0	39.3
	Ours	**89.6**	**80.8**	28.3	**46.6**	**61.3**

< Google Robot SIMPLER 결과 >

CogACT

- Experiment – Simulation Benchmark
  - WidowX Robot
    - Visual Matching: 현실과 유사한 조건

WidowX Robot	Method	Put Spoon on Towel	Put Carrot on Plate	Stack Green Block on Yellow Block	Put Eggplant in Yellow Basket	*Average*
SIMPLER (Visual Matching)	RT-1-X [48]	0.0	4.2	0.0	0.0	1.1
	Octo-Base [62]	15.8	12.5	0.0	41.7	17.5
	Octo-Small [62]	41.7	8.2	0.0	56.7	26.7
	OpenVLA [30]	4.2	0.0	0.0	12.5	4.2
	Ours	**71.7**	**50.8**	**15.0**	**67.5**	**51.3**

< WidowX Robot SIMPLER 결과 >

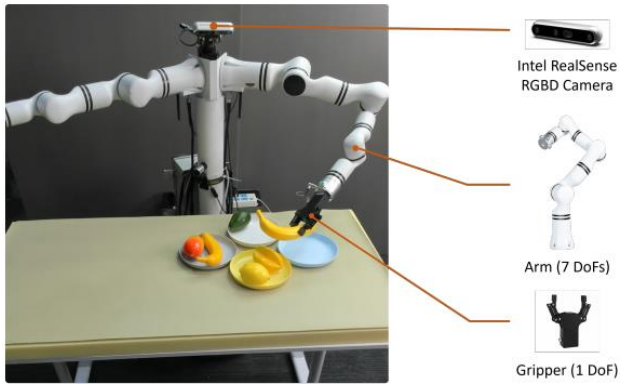
  - Simulation Benchmark 결과 정리
    - WidowX는 Google robot과 다른 형태의 로봇
    - CogACT 평균 성능: 51.3% (OpenVLA 4.2% 대비 + 47.1% 향상)
    - 시뮬레이션 성능이 다양한 로봇 플랫폼에서 일관되게 유지되는지 Generalization 검증

CogACT

- Experiment – Real world Benchmark

Method	Pick				Stack			Place			Task (All)
	Banana	Lemon	Avocado	Avg.	Cup	Bowl	Avg.	Pick	Stack	Avg.	Avg.
Octo-Base [62]	25.0	0.0	0.0	8.3	0.0	0.0	0.0	12.5	0.0	6.3	4.9
OpenVLA [30]	12.5	12.5	0.0	8.3	25.0	6.3	15.6	25.0	4.2	12.5	12.1
Ours	**75.0**	**50.0**	**87.5**	**70.8**	**95.8**	**68.8**	**82.3**	**87.5**	**33.3**	**60.4**	**71.2**

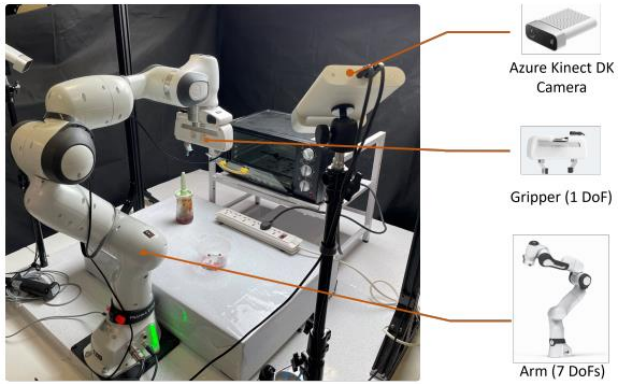
< Realman Robot 결과 >



< Realman Robot 사진 >

Method	Close Oven	Open Oven	Pick Bowl	Pick Brush	Avg.
Octo-Base [62]	0.0	0.0	27.3	0.0	5.8
OpenVLA [30]	18.2	0.0	9.1	0.0	6.8
Ours	**63.6**	**72.7**	**72.7**	**36.4**	**61.4**

< Franka Robot 결과 >



< Franka Robot 사진 >

- Real world Benchmark 결과 정리
  - SIMPLER Simulation 성능이 실제 로봇에서도 재현
  - Realman Robot CogACT 평균 성능: 71.2% (OpenVLA 12.1% 대비 + 59.1% 향상)
  - Franka Robot CogACT 평균 성능: 61.4% (OpenVLA 6.8% 대비 + 54.6% 향상)

CogACT

- Ablation study – Architecture & Multi-step Design & Adaptive Action Ensemble

▪ Action Model Architecture

- Transformer 백본이 필수적
- DiT-Large (308M): 64.8% 평균
- MLP 대비 14.2% 향상 (50.6% → 64.8%)
- 모델 크기 증가에 따른 선형적 성능 향상 가능성

Action Model	Params	GR (VM)	GR (VA)	WR (VM)	Average
MLP (3-Layer)	3M	52.2	52.4	47.1	50.6
MLP (7-Layer)	89M	61.4	48.0	48.1	52.5
DiT-Small	13M	73.3	51.3	51.0	58.5
DiT-Base	89M	74.8	**61.3**	51.3	62.5
DiT-Large	308M	**76.7**	59.3	**58.3**	**64.8**

< Architecture 비교 >

▪ Multi-step Action Prediction

- 현재 시점뿐 아니라 미래 N개의 step을 예측
- 최적값: $N = 15$ (62.5%)
- $N = 0$ (single-step): 42.8% → 19.7% 향상
- $N = 31$과 같은 과도한 예측은 성능 저하

Future Steps	GR (VM)	GR (VA)	WR (VM)	Average
0	73.4	49.0	6.3	42.8
3	70.4	58.9	37.1	55.5
15	**74.8**	**61.3**	51.3	**62.5**
31	54.3	47.6	**51.7**	51.2

< N Prediction 비교 >

▪ Adaptive Action Ensemble

- 유사도 기반 가중치로 모드 혼합 방지
- 최적값: AAE (62.5%)
- Temporal Ensemble 58.9% → 3.6% 향상

Strategy	GR (VM)	GR (VA)	WR (VM)	Average
Action Chunking	67.4	52.5	32.1	50.7
Temporal Ensemble	**75.0**	59.9	41.9	58.9
Adaptive Ensemble	74.8	**61.3**	**51.3**	**62.5**

< Ensemble Strategy 비교 >

※ GR: Google Robot / WR: WidowX Robot

CogACT

- Qualitative Results

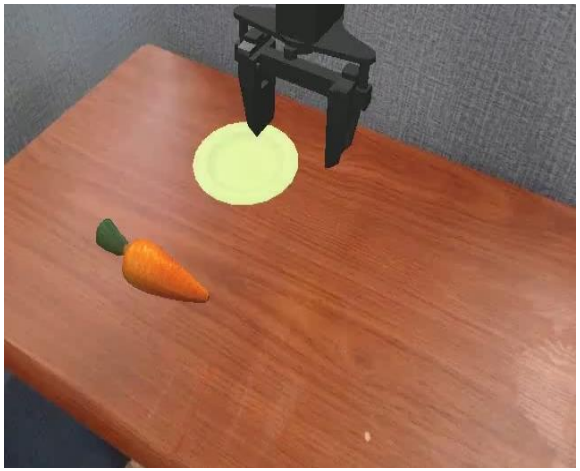
Pick up the screwdrive
Put the screwdriver into the basket

< Realman Robot >

Open the oven door

< Franka Robot >

Put carrot on plate

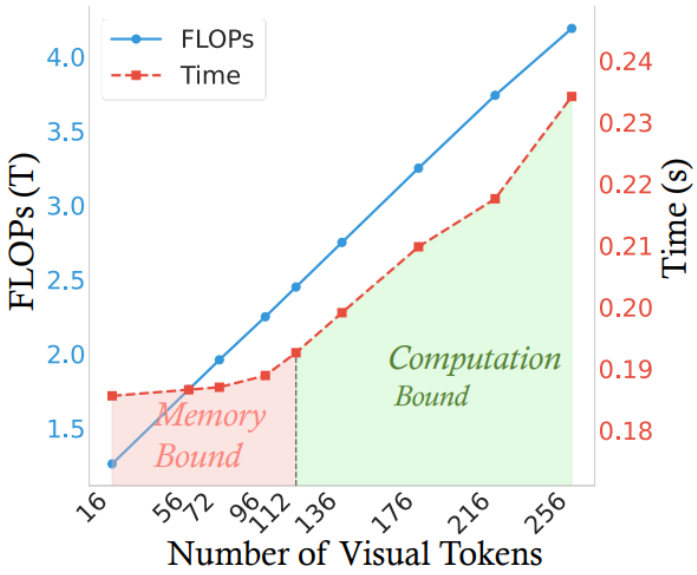


< WidowX Robot >

EfficientVLA: Training-Free Acceleration and Compression for Vision-Language-Action models [NeurIPS 2025]

EfficientVLA

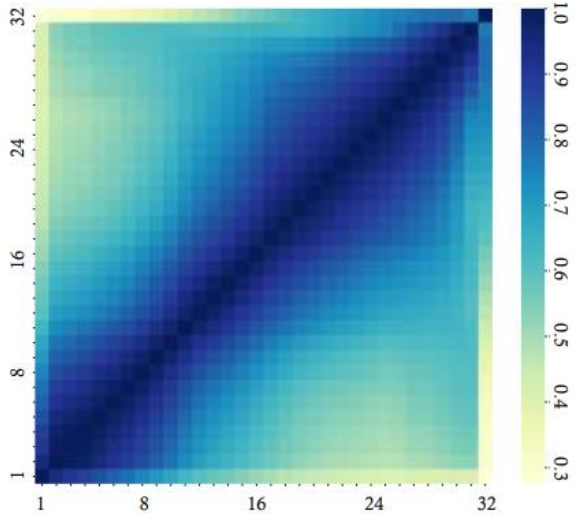
- Introduction
  - Problem 1: High Computational Cost
    - Diffusion-based VLA는 10개의 denoising steps 반복
    - 각 step마다 256개의 Visual Token x LLM 연산이 반복
  - Problem 2: Memory Bottleneck
    - 파라미터: 6.7B → Memory: ~13GB
    - 처리: 256 tokens * 32 layers
    - 결과: VRAM 포화 → Edge Device 불가능
  - Problem 3: Token-Pruning의 한계
    - Computational Bound 구간에서는 FLOPs와 Time이 비례
    - Memory Bound 구간에서는 계산 최적화의 효과가 거의 없음
    - 추가적인 가속을 위해 Language Module, Temporal Caching등 추가 경량화 적용 필요



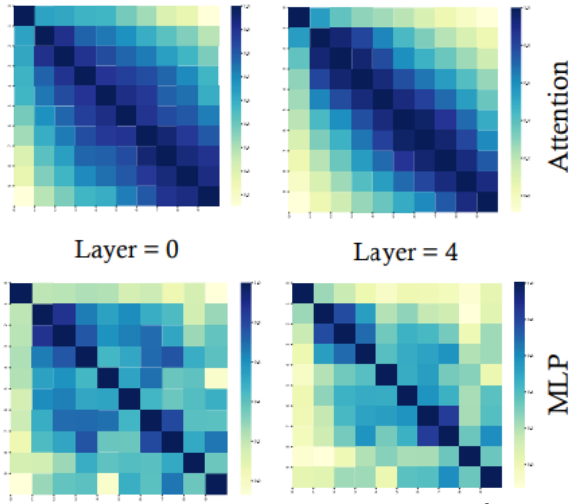
< Token-Pruning 한계 시각화 >

EfficientVLA

- Introduction
  - Depth-wise Redundancy (Language Module)
    - LLM의 깊은 레이어에서 입출력이 거의 동일
    - Cosine Similarity: 0.8~0.95로 매우 높음
  - Visual Token Redundancy
    - 256개의 Token 중 대부분이 배경 & 반복 정보
    - 로봇 제어에 필수적인 팔 + 물체 + Gripper만 필요
    - 약 80% 이상의 Visual Token이 불필요
  - Temporal Redundancy (Diffusion)
    - Denoising step간 중간 특징이 거의 동일
    - 인접한 Denoising Step간의 결과가 거의 동일



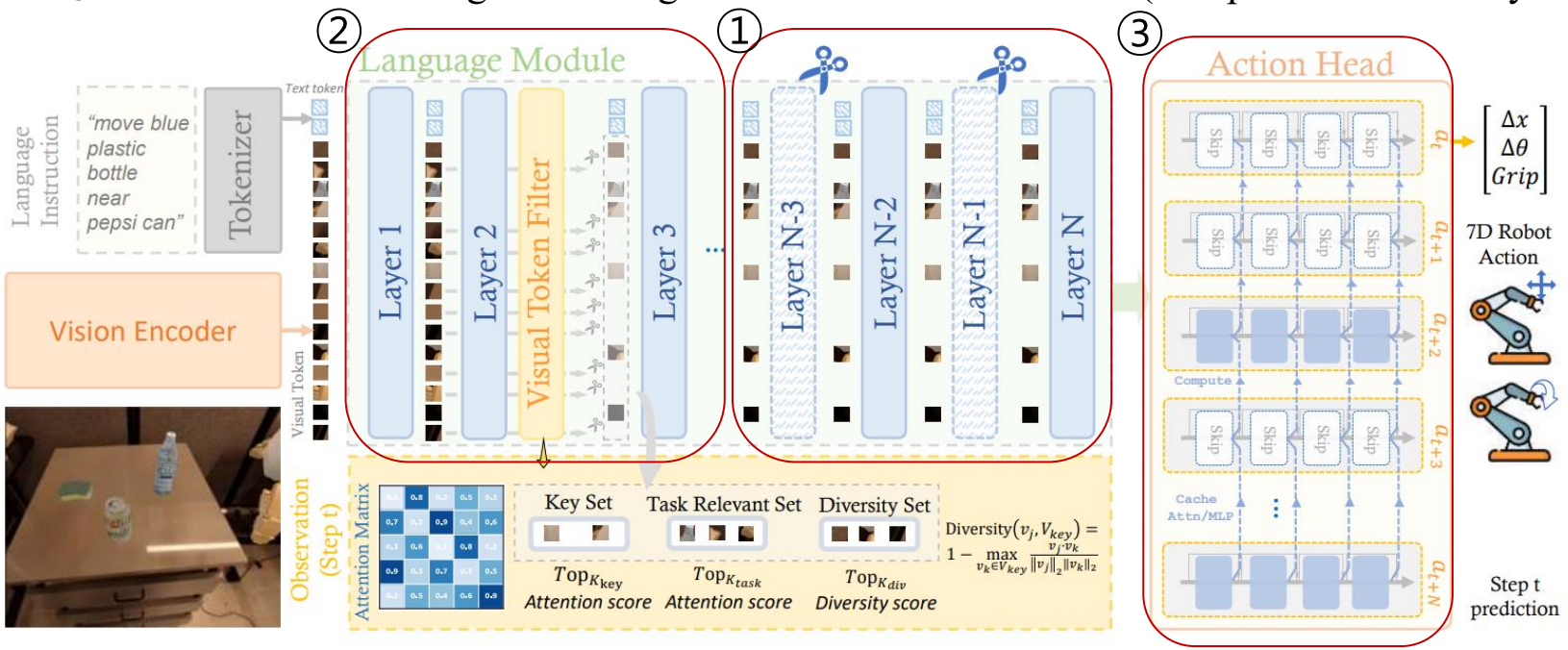
< Layer-wise Cosine 유사도 >



< Timestep-wise Cosine 유사도 >

EfficientVLA

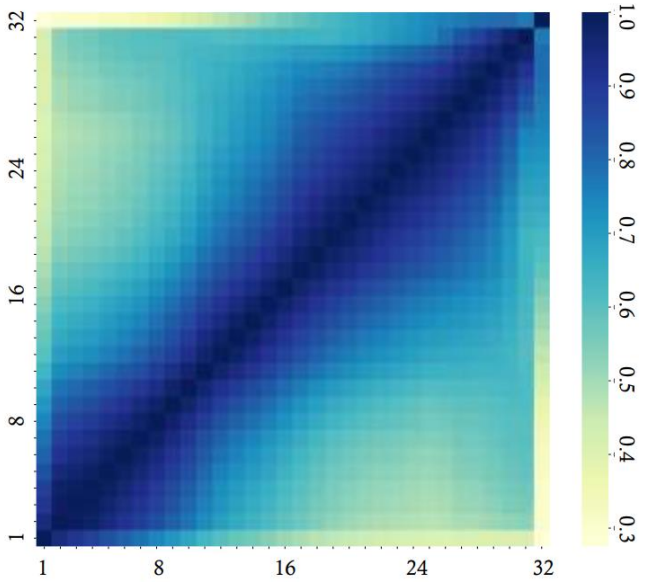
- Overall Framework
  - VLA 모델의 3가지 모듈 각각에 존재하는 Redundancy를 제거한 통합 Framework
    - ① LLM Layer Pruning: 중요도가 낮은 LLM Layer 제거 (Depth-wise Redundancy 해결)
    - ② Token Pruning: Task 관련성과 다양성을 고려한 Token 선별 (Spatial Redundancy 해결)
    - ③ Action Head Caching: Denoising 단계의 중간 특징값 캐싱 (Temporal Redundancy 해결)



< EfficientVLA Framework >

EfficientVLA

- Method 1: LLM Layer Pruning
  - Observation
    - 깊은 레이어로 갈수록 입/출력 간 Cosine Similarity가 매우 높음
    - 해당 레이어가 표현 변환에 기여하는 바가 적음을 의미
  - Importance Score (I^{ℓ})
    - $I^{\ell} = 1 - AvgCosineSim\left(Input\left(x^{\ell}\right), Output\left(x^{\ell+1}\right)\right)$
    - 유사도가 높을수록 Layer 중요도 점수가 낮아짐
  - Pruning Strategy
    - Importance Score가 가장 낮은 N개의 Layer를 Pruning
    - 비연속적으로 불필요한 레이어만 골라냄



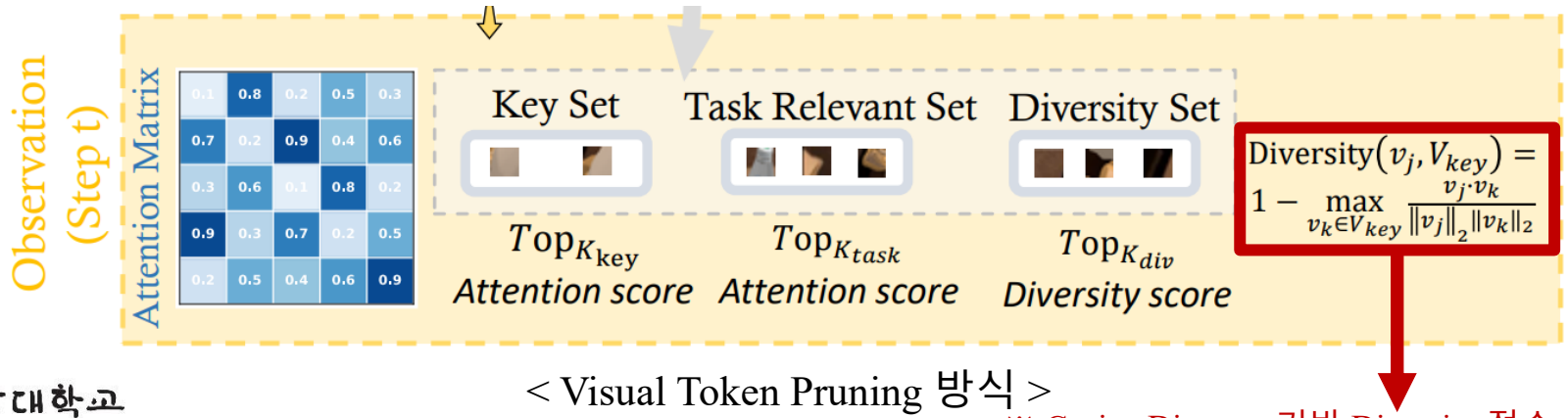
< Layer-wise Cosine 유사도 >

EfficientVLA

- Method 2: Visual Token Pruning
  - Challenge
    - 무작위로 Visual Token을 Pruning하면 로봇 제어에 필요한 핵심 정보가 소실됨
  - Selection Criteria
    - Key Set: Text Prompt와 Cross-Attention이 가장 높은 핵심 토큰 선발
    - Task Relevant Set: 남은 Token중 Attention 점수가 높은 순으로 추가
    - Diversity Set
      - 기존 선택된 Token들과 유사도가 낮은 Token을 선택
      - 배경 정보나 주변 맥락을 보존하여 Visual 정보의 편향을 방지하기 위함

$$r_i = \sum_{j=1}^{L_{ctx}} \left(\frac{1}{H} \sum_{h=1}^{H} A_{i,j}^{(h)} \right)$$

$$r_i = \sum_{j} \mathrm{Avg}(A_{i,j})$$

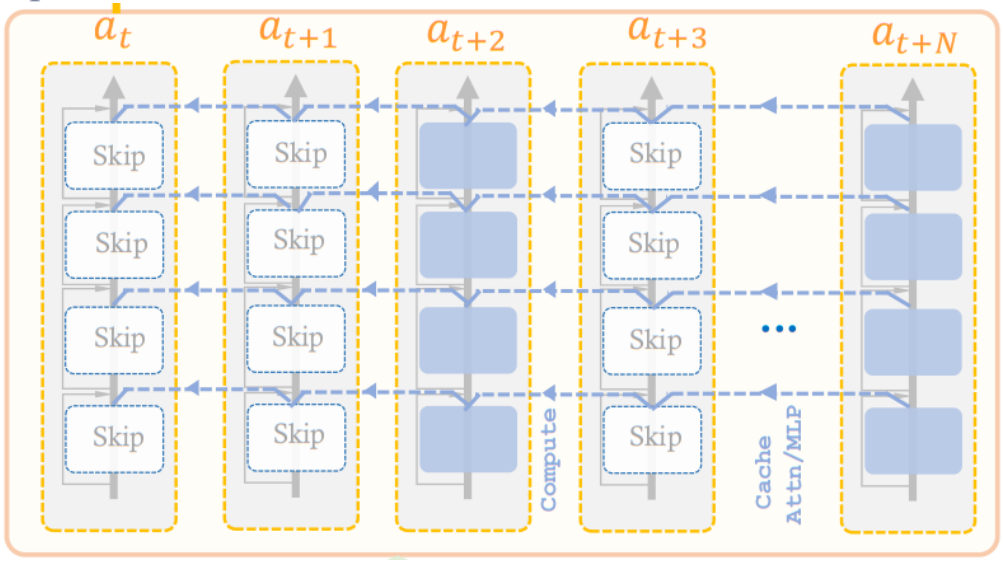


< Visual Token Pruning 방식 >

※ Cosine Distance 기반 Diversity 점수 산출 방식

EfficientVLA

- Method 3: Action Head Caching
  - Observation
    - Diffusion의 Denoising Step간 Attention 및 MLP의 중간 출력값 변화가 거의 없음
  - Caching Strategy
    - 매 Step마다 무거운 연산을 반복하지 않음
    - K Step 간격으로만 연산을 수행하고, 그 사이 단계에서는 캐싱된 값을 그대로 재사용
      - K번째 Step 마다 한 번씩만 계산 및 캐싱하고, 나머지 4번은 캐싱된 값을 재사용



< Action head Caching 방식 >

※ $K = 5$, 논문 기본값

EfficientVLA

- Experiment – Simulation Benchmark
  - SIMPLER 환경에서 CogACT(baseline)에 EfficientVLA를 적용해 성능을 평가
    - Inference Time: CogACT(baseline) 대비 1.93배 빠른 추론 속도
    - Computae: FLOPs를 baseline 대비 28.9% 수준까지 감소 (71.1% 감소)
    - Accuracy: 평균 success rate는 0.6% 수준의 최소 감소

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

< SIMPLER: CogACT vs EfficientVLA >

※ L은 Layer 개수, T는 Token 개수

EfficientVLA

- Ablation Study
  - Token Pruning만 적용시 ~1.23x에 머물러 LLM Memory/action head 병목 남음
  - Model Compression만 적용해도 ~1.43x까지 증가하지만 성능 저하 발생
  - 모든 경량화 기법을 적용하여 측정시 추론 속도가 ~1.93x까지 증가
  - 불필요한 Visual Token제거 → Hallucination 감소 → Success Rate ↑

	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**×

< 단일 task에서의 Ablation study >

Conclusion

- CogACT
  - 기존 VLA(Gen 1)의 한계점인 Cognitive Gap을 Cognition-Action 분리 구조로 해결
  - LLM의 추론 능력과 Diffusion Action Model의 정교한 제어 능력을 결합하여 Generalization 및 Precision 확보

- EfficientVLA
  - CogACT와 같은 고성능 VLA 모델의 Redundancy (Visual, Layer, Temporal)를 체계적으로 분석
  - Training-free 방식의 최적화를 통해 성능 저하 없이 1.93배 가속 & 메모리 효율성 달성

- Final Thought
  - High-Performance Reasoning과 Deployment Efficiency를 동시에 달성하여, 실제 로봇 환경에서도 VLA의 상용화 가능성을 입증한 논문

감사합니다