2026 여름 세미나

2026.07.24

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

Presented By
이준호

Outline

- Background
  - Vision-Language-Action (VLA) model
  - Model Merging
- Paper 1
  - VLA-Adapter: An Effective Paradigm for Tiny-Scale Vision-Language-Action Model (AAAI 2026)
- Paper 2
  - MergeVLA: Cross-Skill Model Merging Toward a Generalist Vision-Language-Action Agent (CVPR 2026)

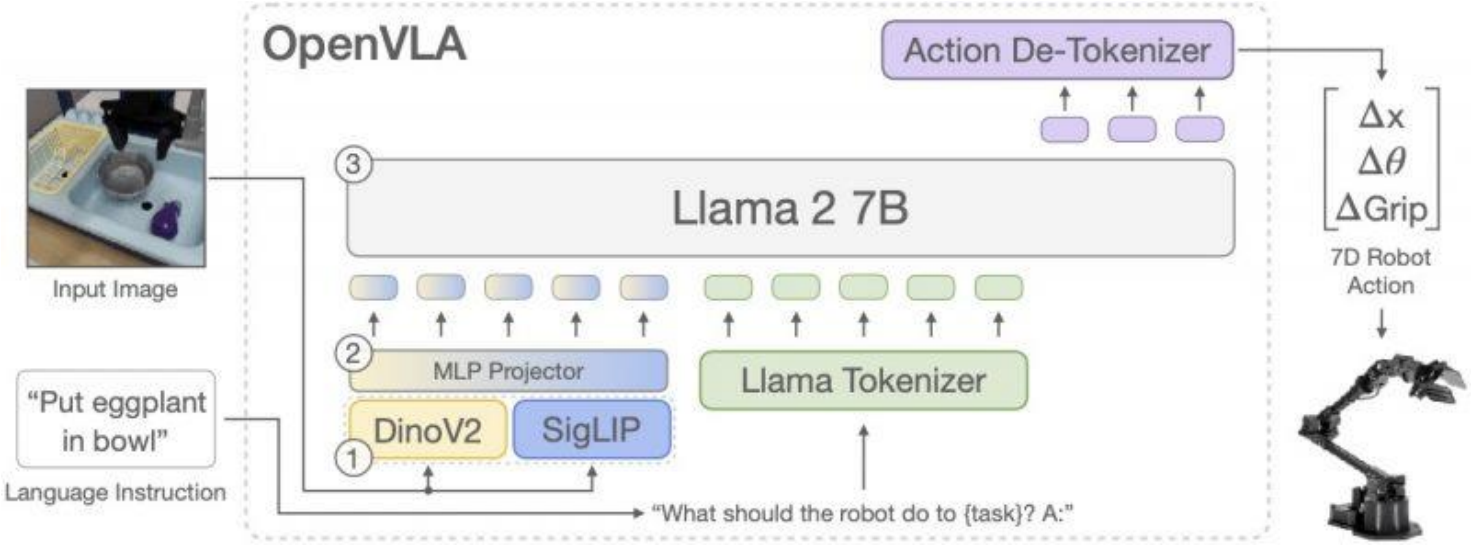
Background

- Vision-Language-Action (VLA) 모델
  - 대규모 로봇 조작 데이터셋으로 VLM을 사전학습 시킨 뒤 policy 네트워크에 전달하여 다양한 환경의 task에 맞는 action을 생성하도록 설계됨
  - 멀티모달 정보를 추출하여 이를 action 공간과 정렬시킴으로써 고품질의 행동을 생성하는 데 초점을 맞추고 있음
    - 대표 모델: OpenVLA, π0 / π0.5 / π0.7, VLA-Adapter 등
  - Single task 및 Single embodiment에서 뛰어난 성능을 보임

- VLA 모델의 발전 흐름
  - 더 큰 사전학습 지식을 물려받으면서도, 더 가볍고 빠른 action 생성 구조로 진화
    - **2023 · RT-1 / RT-2**: 웹 지식을 로봇 제어로 전이, action을 이산 토큰으로 생성
    - **2024 · OpenVLA**: 오픈소스 7B VLA, action을 어휘 토큰으로 이산화·자기회귀 생성
    - **2024 · π0 / π0.5**: dual-system 구조 + flow matching으로 연속 action chunk를 빠르게 생성
    - **2025 · VLA-Adapter 등**: 소형(0.5B) backbone + 별도 action expert로 경량·고성능 달성

Background

- Model Merging
  - 여러 finetuning 모델의 weight를 재학습·원본 데이터 재접근 없이 결합
  - LLM·VLM 도메인에서는 이미 효과가 입증된 기법
    - 대표 기법: Task Arithmetic (TA), TIES-Merging 등
  - 목표: $\{\Theta_1, \ldots, \Theta_M\} \rightarrow$ 하나의 범용 정책 Θ_{merge}



VLA-Adapter: An Effective Paradigm for Tiny-Scale Vision-Language-Action Model (AAAI 2026)

Paper 1

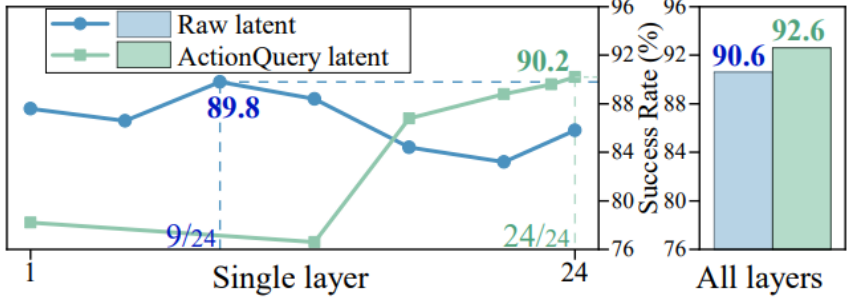
- Contribution
  - 소형 backbone만으로 SOTA급 VLA 성능을 달성하는 경량 bridging 메커니즘 제안
    - **체계적 분석:** Vision-Language 표현을 Action으로 연결하는 다양한 조건 (raw feature vs additional query, layer 위치)의 효과를 VLA 분야 최초로 분석하고 핵심 설계 원칙 도출
    - **Bridge Attention 설계:** VLM 전체 layer의 Raw latent와 ActionQuery latent를 학습 가능한 비율로 융합해 action space에 주입하는 경량 Policy(97M) 제안
    - **효율성과 성능 동시 확보:** 0.5B backbone만으로 OpenVLA-OFT(7B) 대비 파라미터 1/14, 학습 VRAM 0.4배, 추론 속도 3배를 달성하면서 LIBERO·CALVIN·실제 로봇에서 동등 이상의 성능 확보

	OpenVLA-OFT (SOTA)	VLA-Adapter (Ours)	
Backbone ↓	7B	0.5B	1/14×
Training VRAM (8 batch) ↓	62GB	24.7GB	0.4×
Throughput (8-dim chunk) ↑	71.4Hz	219.2Hz	3×
Performance (LIBERO) ↑	97.1%	97.3%	Maintain

Paper 1

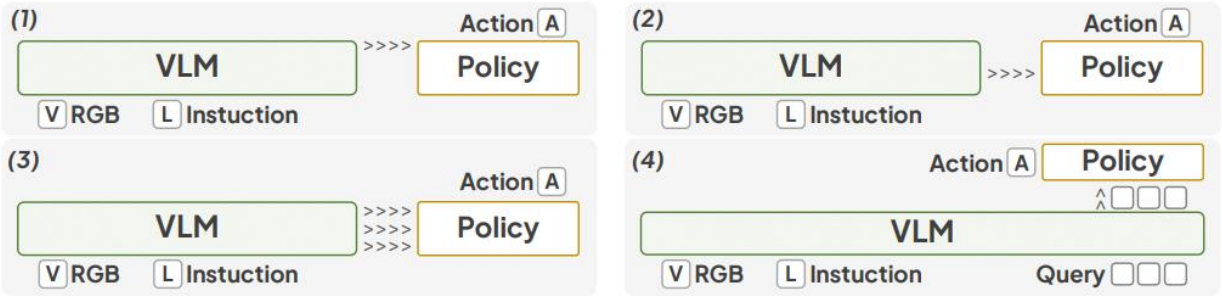
- Motivation 1
  - 기존 VLA 모델은 대규모 VLM 사전학습에 강하게 의존
    - 학습 비용·배포 장벽이 큼
    - **표준 구조**:
      - VLM을 대규모 로봇 시연 데이터로 사전학습한 뒤 Policy 네트워크에 전달하여 다양한 환경의 과제에 맞는 행동을 생성하도록 설계
        - ✓대형 backbone(7B+)에 대한 높은 의존성, 과도한 VRAM, 느린 추론이 필연적으로 수반됨
    - **핵심 문제 정의**:
      - "VL 표현을 A로 어떻게 더 효과적으로 bridging할 것인가?"는 VLA 설계의 가장 근본적인 문제이지만 충분히 논의되지 않음
    - **VLA-Adapter의 목표**: 대규모 로봇 데이터 사전학습 없이도 tiny-scale(0.5B) backbone만으로 SOTA급 성능 달성

Paper 1



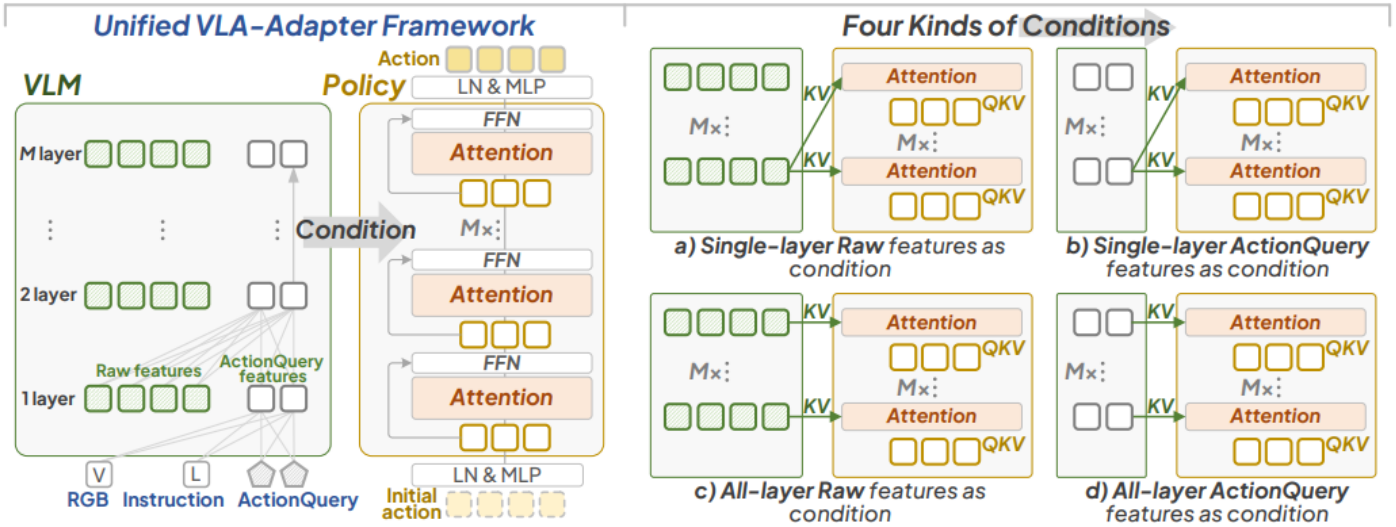
- Motivation 2
  - 기존 bridging 방식들이 제각각이라 어떤 조건이 실제로 중요한지 불분명함
    - 4가지 대표 패러다임: (1)RoboVLMs–마지막 layer raw, (2)GR00T N1–중간 layer raw, (3)π0–전체 layer raw, (4)OpenVLA-OFT–마지막 layer의 additional query
    - 문제: **layer 위치**와 **feature 종류**가 뒤섞여 있어 실제 성능 기여 요인을 알 수 없음
      - Key Finding 1: Raw latent는 middle layer가 deep layer보다 효과적
      - Key Finding 2: ActionQuery latent는 처음부터 학습되므로 deep layer일수록 효과적
      - Key Finding 3: 단일 layer보다 all-layer 사용이 항상 우수 (성능↑, layer 선택 불필요)

Type	Representative work	Feature type	Which layer
(1)	RoboVLMs	Raw features	Last
(2)	GR00T N1	Raw features	Intermidiate
(3)	π_0	Raw features	All
(4)	OpenVLA-OFT	Additional Query	Last



Paper 1

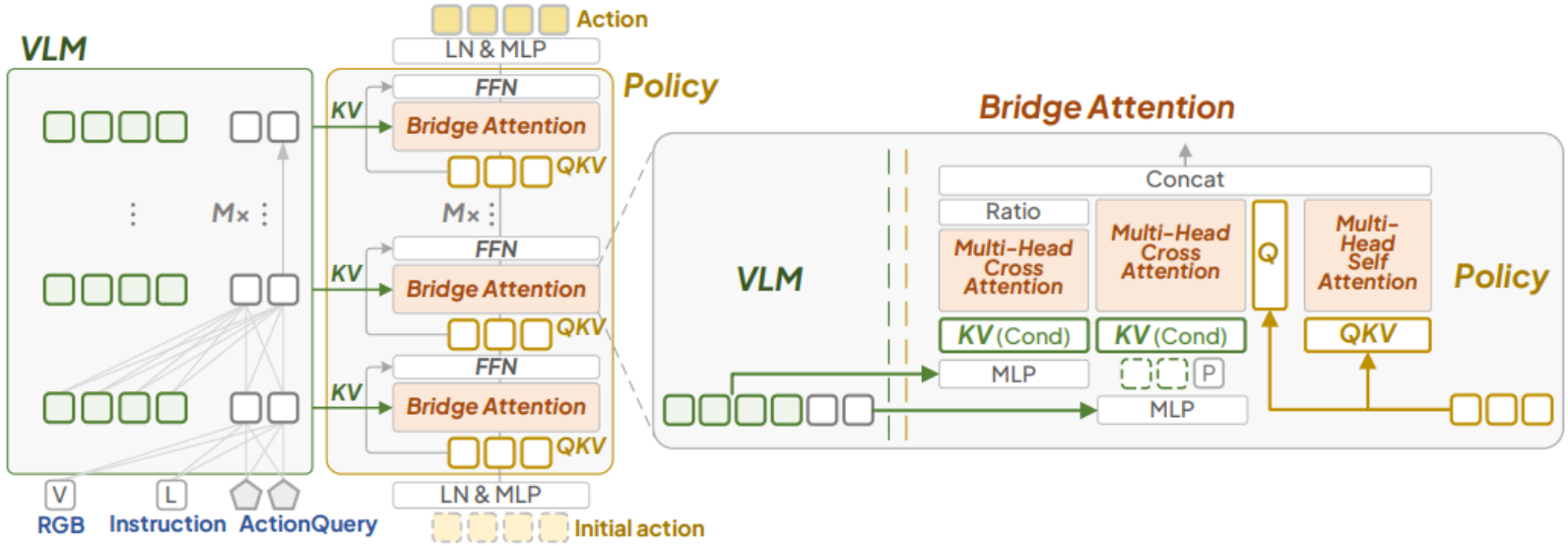
- Method Overview
  - VLM의 모든 layer에서 두 종류의 조건을 동시에 추출해 Policy에 주입하는 Unified 프레임워크
    - **Raw latent:** DINOv2/SigLIP 비전 임베딩과 언어 임베딩이 결합된 VLM 내부 표현
    - **ActionQuery latent:** 학습 가능한 query 토큰이 VLM을 통과하며 얻는 표현
    - **구조:** Policy는 VLM과 동일한 M개 layer로 구성, 각 layer가 해당 layer의 두 조건을 사용 (Key Finding 3 반영)
    - **Condition Determination:** All-layer ActionQuery만으로도 우수하지만 일부 hard task는 middle-layer Raw feature가 더 강함 → 두 조건을 함께 사용



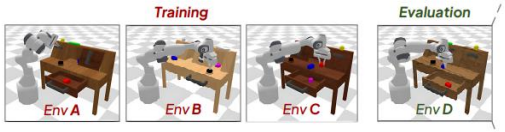
Paper 1

극단적 값으로 인한 학습 불안정 방지

- Method – Bridge Attention
  - 두 조건을 학습 가능한 비율로 융합하는 경량 attention 모듈 (Parameter: 97M)
    - **구성:** 각 Policy layer = Bridge Attention + FFN
    - **CA1:** action latent(Q) ↔ Raw latent(K,V) + learnable ratio (tanh)로 주입 정도 조절
    - **CA2:** action latent(Q) ↔ [ActionQuery latent, proprioceptive state](K,V)
    - **SA + 학습:** 3가지 attention 결과를 concat, L1 loss로 end-to-end 학습
      - L1 loss가 추론 속도면에서 우수하여 채택

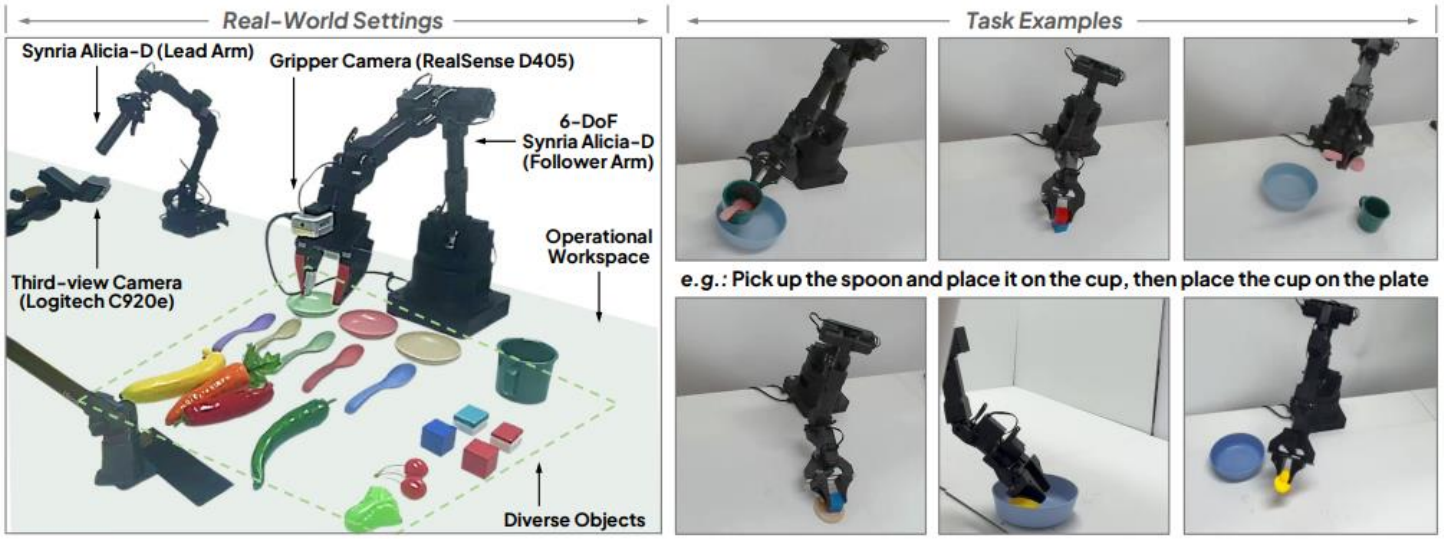


Paper 1



Task instruction	Condition
1. Rotate {object} right/left	[rotate_object, '{object}', ±60]
2. Push {object} right/left	[push_object, '{object}', ±0.1, 0]
3. Move slider right/left	[move_door_rel, 'base_slide', ±0.23]
Open/Close drawer	[move_door_rel, 'base_drawer', ±0.12]
4. Lift {object} table/drawer	[lift_object, '{object}', 0.05, 'table', 'base_link/drawer_link']
Lift {object} slider	[lift_object, '{object}', 0.03, 'table', 'plank_link']
5. Place in slider/drawer	[stack_objects/unstack_objects]
6. Stack/Unstack block	[place_object, 'table', 'plank_link/drawer_link']
7. Turn on/off lightbulb/led	[toggle_light, 'lightbulb/led', 0/1, 1/0]
8. Push into drawer	[push_object_into, [{object}], 'table', 'base_link', 'table', 'drawer_link']

- Experiments Setup
  - LIBERO / CALVIN ABC→D / 실제 로봇, 3개 환경에서 종합 검증
    - **LIBERO:** Spatial/Object/Goal/Long 4개 suite × 10 task, task당 50회 시행 (성능 비교 중심)
    - **CALVIN ABC→D:** Env A,B,C로 학습 후 Env D에서 zero-shot 평가, 5-task 연쇄 수행 (일반화 능력 검증)
    - **실제 로봇:** 6-DoF Synria Alicia-D + Logitech/RealSense 카메라, pick·move·stack·long-horizon 4개 task, 매 실행마다 물체 위치 랜덤화
    - **구현:** backbone Qwen2.5-0.5B + LoRA + AdamW, batch size 16, 4×H100 GPU (Single GPU 기준 8시간 학습 완료 가능)



Paper 1

Frozen	OpenVLA-OFT	SmolVLA	**VLA-Adapter**
Success Rate (%) ↑	0.0	77.0	**86.4**

- Results – Necessity & Efficiency
  - backbone 규모·사전학습 여부와 무관하게 안정적으로 작동, 추론 속도도 가장 빠름
    - **Fine-tuned 비교 (LIBERO-Long Success Rate, OFT 대비 Ours):**
    - **Frozen backbone:** OpenVLA-OFT 0.0% < SmolVLA 77.0% < VLA-Adapter 86.4%
    - **Efficiency:** Throughput 71.4→219.2 Hz (3배↑), Latency 0.112→0.0365 sec
    - 사전학습되지 않은 VLM일수록 이득이 크고, backbone이 frozen되어도 강한 성능 유지

Backbone	**+OFT**	**+Ours (VLA-Adapter)**
Qwen2.5-0.5B	85.8	**95.0 (+9.2%p)**
LLaMA2-7B	87.5	**95.2 (+7.7%p)**
OpenVLA-7B (pretrained)	94.5	**95.4 (+0.9%p)**

Efficiency	OpenVLA	OpenVLA-OFT (wo $\mathcal{X}_t^g$, $\mathcal{P}$)	OpenVLA-OFT	**VLA-Adapter**
Throughput (Hz) ↑	4.2	109.7	71.4	**219.2**
Latency (Sec) ↓	0.2396	0.0729	0.1120	**0.0365**

Paper 1

- Results – LIBERO Benchmark
  - 0.5B backbone만으로 7B급 SOTA(OpenVLA-OFT)와 동등하거나 상회하는 성능
    - 동일 tiny-scale(0.5B)에서 VLA-OS보다 LIBERO-Long +29.0%, OpenVLA-OFT(7B) 대비 파라미터 1/14 · VRAM 0.4배 · throughput 3배로 동등 이상 성능 확보

	LIBERO	Params	Spatial	Object	Goal	Long	Avg.
Large	FlowVLA (Zhong et al., 2025) *(ArXiv)*	8.5	93.2	95.0	91.6	72.6	88.1
	UnifiedVLA (Li et al., 2025a) *(ArXiv)*	8.5	95.4	*98.8*	93.6	94.0	95.5
	OpenVLA (Kim et al., 2024) *(CoRL)*	7	84.7	88.4	79.2	53.7	76.5
	OpenVLA-OFT (Kim et al., 2025) *(RSS)*	7	*97.6*	98.4	**97.9**	*94.5*	*97.1*
	UniVLA (Bu et al., 2025) *(RSS)*	7	96.5	96.8	95.6	92.0	95.2
	CoT-VLA (Zhao et al., 2025a) *(CVPR)*	7	87.5	91.6	87.6	69.0	81.1
	WorldVLA (Cen et al., 2025) *(ArXiv)*	7	87.6	96.2	83.4	60.0	81.8
	TraceVLA (Zheng et al., 2024) *(ArXiv)*	7	84.6	85.2	75.1	54.1	74.8
	MolmoAct (Lee et al., 2025) *(ArXiv)*	7	87.0	95.4	87.6	77.2	86.6
	ThinkAct (Huang et al., 2025) *(ArXiv)*	7	88.3	91.4	87.1	70.9	84.4
	PD-VLA (Song et al., 2025b) *(ArXiv)*	7	95.5	96.7	94.9	91.7	94.7
Small	4D-VLA (Zhang et al., 2025a) *(ArXiv)*	4	88.9	95.2	90.9	79.1	88.6
	SpatialVLA (Qu et al., 2025) *(RSS)*	4	88.2	89.9	78.6	55.5	78.1
	π_0 (Black et al., 2025b) *(RSS)*	3	96.8	*98.8*	95.8	85.2	94.2
	π_0-FAST (Pertsch et al., 2025) *(RSS)*	3	96.4	96.8	88.6	60.2	85.5
	NORA (Hung et al., 2025) *(ArXiv)*	3	92.2	95.4	89.4	74.6	87.9
	SmolVLA (Shukor et al., 2025) *(ArXiv)*	2.2	93.0	94.0	91.0	77.0	88.8
	GR00T N1 (NVIDIA et al., 2025) *(ArXiv)*	2	94.4	97.6	93.0	90.6	93.9
	GraspVLA (Deng et al., 2025) *(ArXiv)*	1.8	-	94.1	91.2	82.0	89.1
Tiny	Seer† (Tian et al., 2025) (Scratch) *(ICLR)*	0.57	-	-	-	78.7	78.7
	VLA-OS (Gao et al., 2025) *(ArXiv)*	0.5	87.0	96.5	92.7	66.0	85.6
	Diffusion Policy† (Chi et al., 2023) *(RSS)*	-	78.3	92.5	68.3	50.5	72.4
	VLA-Adapter (Ours)	**0.5**	**97.8**	**99.2**	*97.2*	**95.0**	**97.3**
	VLA-Adapter-Pro (Ours)	**0.5**	**99.6***	**99.6***	**98.2***	**96.4***	**98.5***

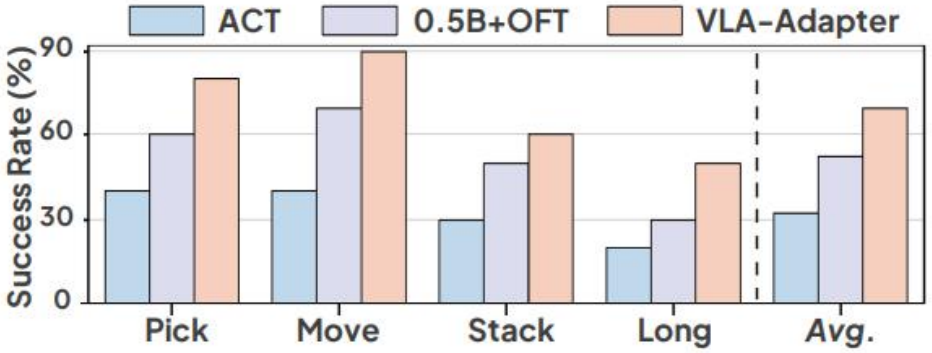
Paper 1

- Results – Generalization & Real-World
  - CALVIN zero-shot 일반화에서 SOTA 수준 유지
    - **CALVIN ABC→D (Avg. task length, 0–5):**
      - Task length: 완료한 task 길이

	CALVIN ABC→D	Params	Task completed in a row ↑ 1	2	3	4	5	Avg. len ↑
Large	UniVLA (Bu et al., 2025) *(RSS)*	7	95.5	85.8	75.4	66.9	56.5	3.80
	OpenVLA (Kim et al., 2024) *(CoRL)*	7	91.3	77.8	62.0	52.1	43.5	3.27
	OpenVLA-OFT (Kim et al., 2025) *(RSS)*	7	96.3	89.1	82.4	75.8	66.5	4.10
	VLAS (Zhao et al., 2025b) *(ICLR)*	7	87.2	64.2	40.9	28.1	19.6	2.40
	LCB (Shentu et al., 2024) *(IROS)*	7	73.6	50.2	28.5	16.0	9.9	1.78
	RoboDual (Bu et al., 2024a) *(ArXiv)*	7	94.4	82.7	72.1	62.4	54.4	3.66
	OpenHelix (Cui et al., 2025) *(ArXiv)*	7	*97.1*	91.4	82.8	72.6	64.1	4.08
	ReconVLA (Song et al., 2025c) *(ArXiv)*	7	95.6	87.6	76.9	69.3	64.1	3.95
Small	DeeR (Yue et al., 2024) *(NeurIPS)*	3	86.2	70.1	51.8	41.5	30.4	2.82
	RoboFlamingo (Li et al., 2024) *(ICLR)*	3	82.4	61.9	46.6	33.1	23.5	2.48
	VPP† (Hu et al., 2025) *(ICML)*	1.5	95.7	91.2	*86.3*	*81.0*	*75.0*	*4.33*
	SuSIE (Black et al., 2024) *(ICLR)*	1.3	87.0	69.0	49.0	38.0	26.0	2.69
Tiny	Seer_{Large}† (Tian et al., 2025) *(ICLR)*	0.57	96.3	*91.6*	86.1	80.3	74.0	4.28
	MoDE† (Reuss et al., 2025) *(ICLR)*	0.44	96.2	88.9	81.1	71.8	63.5	4.01
	Seer† (Tian et al., 2025) *(ICLR)*	0.32	94.4	87.2	79.9	72.2	64.3	3.98
	VLA-Adapter (Ours)	**0.5**	**99.1***	**94.6**	**88.8**	**82.8**	**76.5**	**4.42**
	VLA-Adapter-Pro (Ours)	**0.5**	**98.5**	**95.0***	**90.5***	**85.3***	**80.0***	**4.50***

Paper 1

- Results – Generalization & Real-World
  - 실제 로봇 배치에서도 SOTA 수준 유지
    - **실제 로봇:** Pick·Move·Stack·Long-horizon 4개 suite에서 ACT, 0.5B+OFT 대비 모두 우위
      - 물체 위치 랜덤화로 인한 distribution shift에도 견고



MergeVLA: Cross-Skill Model Merging Toward a Generalist Vision-Language-Action Agent

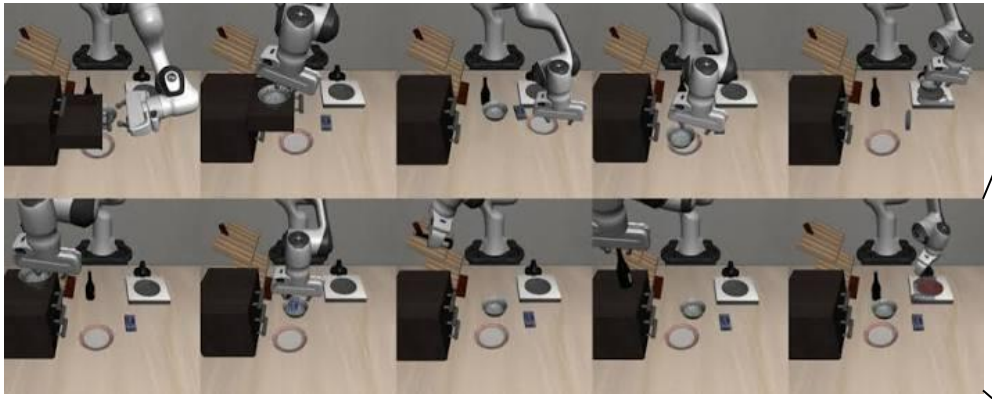
(CVPR 2026)

Paper 2

- Contribution
  - VLA 모델의 task merging 불가능성 원인 규명 및 해결하는 병합 아키텍처 제안
    - **원인 규명**: VLM 백본의 LoRA adapter가 task별로 상충하는 방향으로 발산하는 문제와, action expert의 self-attention이 task 정보를 block 전반에 확산시켜 모듈성을 깨뜨리는 문제를 근본 원인으로 밝혀냄
    - **아키텍처 재설계**: Task mask 기반의 sparse activation LoRA로 VLM 병합 안정화
      - Action expert의 self-attention을 cross-attention 전용 구조로 대체하여 병합 가능성을 설계 단계에서부터 확보함
    - **Task-level Routing**: Task 정체성이 알려지지 않은 상황에서도 value 기반 subspace을 활용해 적절한 task mask와 expert head를 자동 선택하는 training-free router 제안

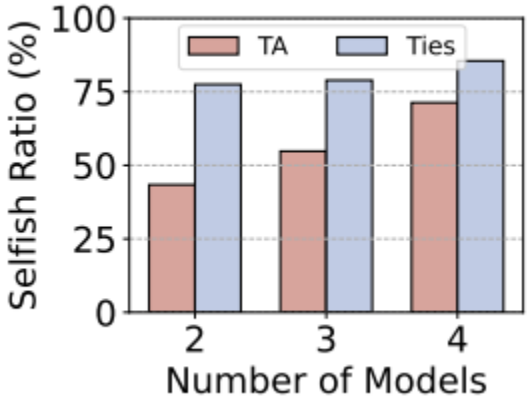
Paper 2

- Motivation 1
  - 서로 다른 skill로 파인튜닝된 VLA expert를 직접 병합하면 성공률이 near-zero (LIBERO 기준 0%)
    - LLM·VLM 도메인에서 검증된 병합 기법(Task Arithmetic, TIES 등)이 VLA에서 무력화됨
    - OpenVLA 자체는 본질적으로 하나의 VLM인데도, 기존 VLM 병합 성공 사례와 상반된 결과를 보임
    - VLA fine-tuning이 task 간 호환 불가능한 **structural specialization**을 유발한다는 것을 시사



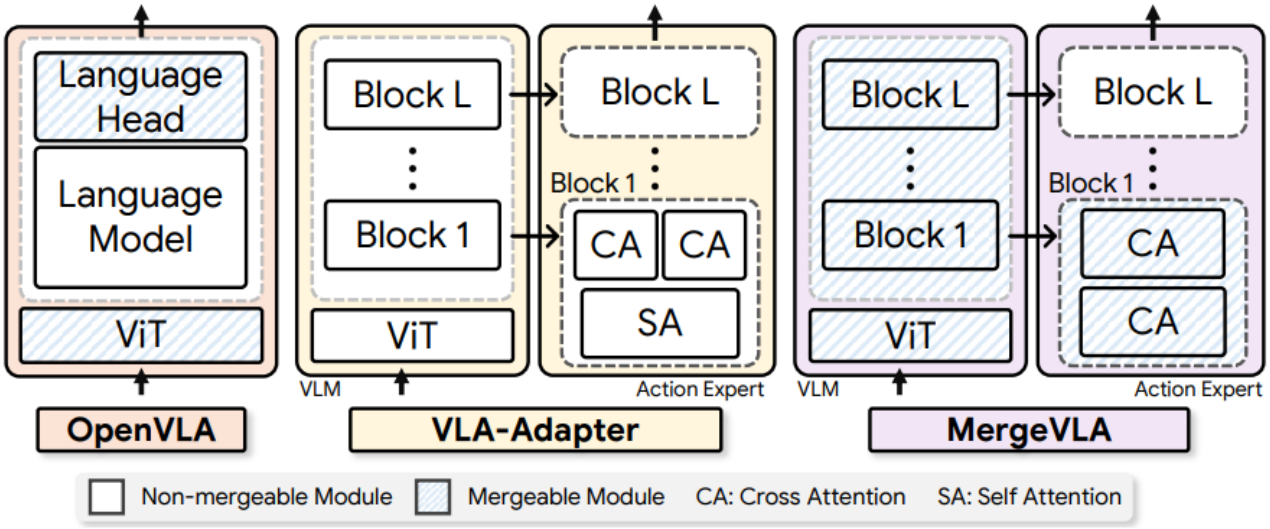
Paper 2

- Motivation 1
  - VLM 백본의 LoRA 업데이트가 task마다 서로 겹치지 않는 파라미터를 활성화하여 task merging 시 파괴적 간섭 발생
    - **Selfish parameter**: 1개의 task에만 유효한 파라미터 비율이, task 4개 병합 시 75%를 초과
    - **원인**: 각 조작 task가 자신의 perception-control alignment에 맞춰 사전학습 표현 공간을 재구성
      - TA나 TIES는 무관·상충 파라미터를 동시에 재활성화
        - ✓공유하는 Vision-Language subspace 훼손
    - **관련 관찰**: ReVLA에서도 VLA fine-tuning이 사전학습 시각 지식을 덮어쓰는 현상을 입증



Paper 2

- Motivation 2
  - VLM을 완벽히 병합해도, action expert를 단순 평균화하면 성공률 0%
    - **원인**: Action expert는 처음부터(from-scratch) 학습되며, self-attention이 task 정보를 block 전반에 전파·증폭
      - 얕은 block은 task 간 정렬을 유지하지만, 깊은 block으로 갈수록 파라미터 거리(L2)가 증가
    - **결과**: 깊은 block들이 개별 task에 특화되어, 어떤 병합 방식으로도 조정 불가능
    - **시사점**: Action expert 자체의 아키텍처 재설계가 필요함



Paper 2

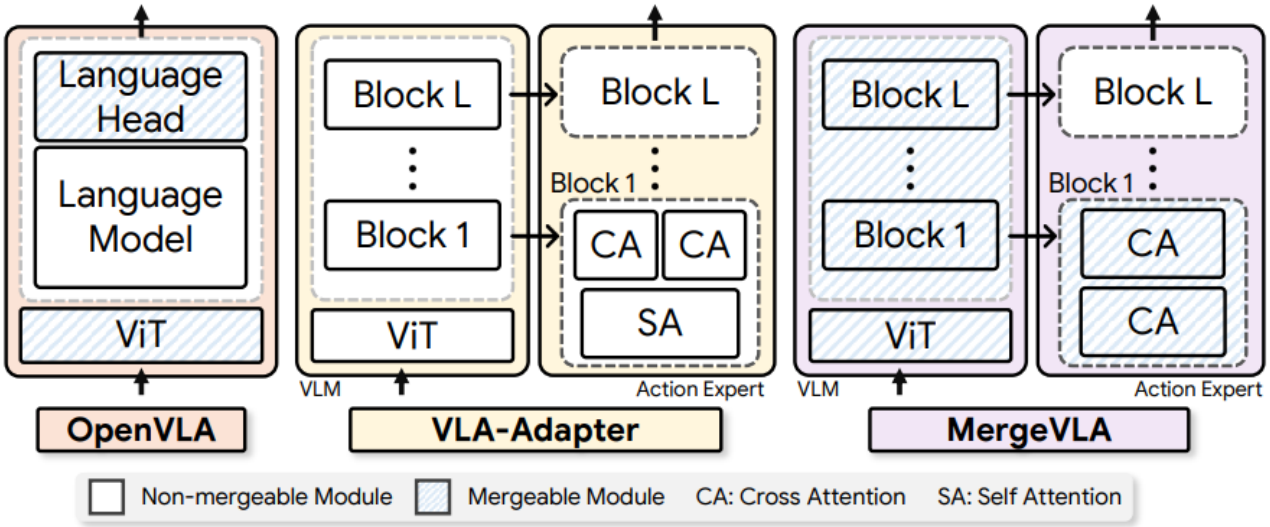
- Motivation 2
  - Self-attention은 이전 block의 action feature를 다시 입력으로 사용함
    - VLA-Adapter의 action expert를 단순화하면 아래와 같음

$$x^{i+1} = SA(x^i) + CA(x^i, h^i_{VLM}) + FFN()$$

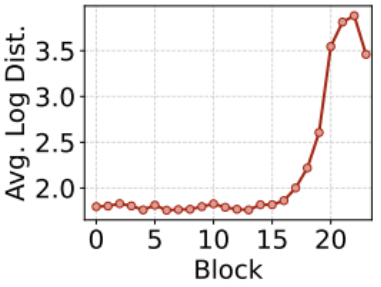
✓여기서 self-attention은 action token x^i끼리 상호작용 함

✓하지만 이 action expert는 pretrain된 모델이 아니기 때문에 각 task별 학습함

✓따라서 각 task의 self-attention은 서로 다른 action-token dependency를 형성함



Paper 2



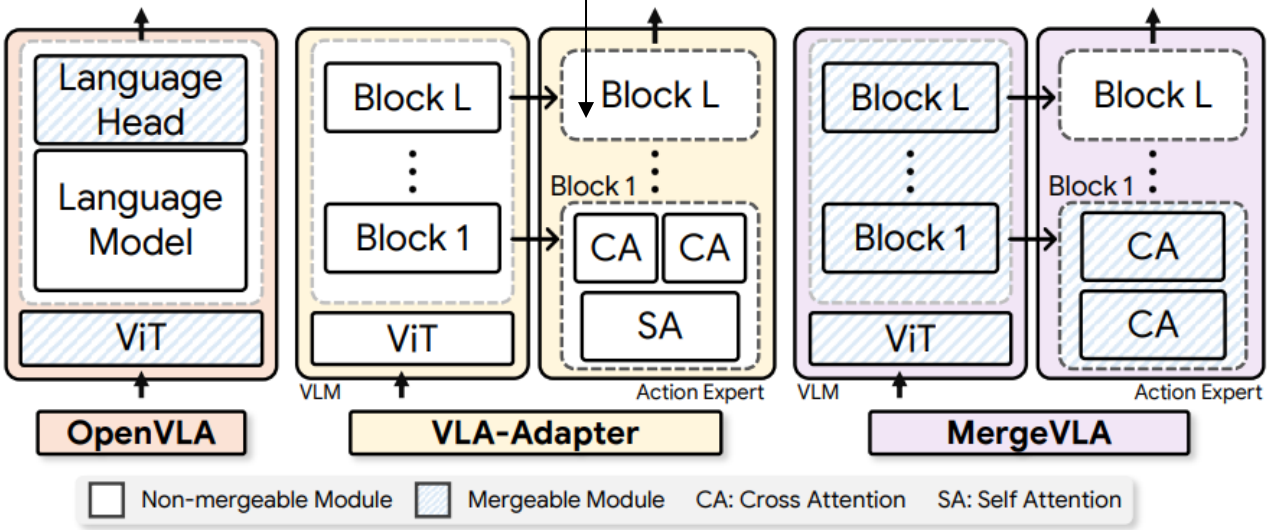
- Motivation 2
  - Self-attention은 이전 block의 action feature를 다시 입력으로 사용함
    - Self-attention이 있는 경우 첫 번째 block에서 생긴 task-specific feature가 다음 block의 self-attention 입력으로 다시 들어감

$$x_{task}^{1} \rightarrow x_{task}^{2} \rightarrow x_{task}^{3} \rightarrow \cdots \rightarrow x_{task}^{L}$$

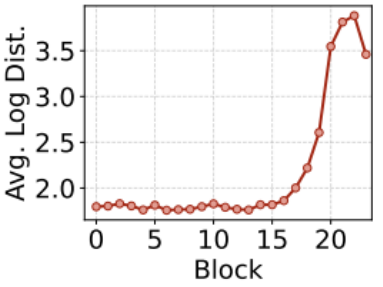
✓즉, 각 block이 독립적인 변환을 수행하는 것이 아니라, 이전 block에서 만들어진 task-specific feature에 의존

✓예를 들어, block 1에서 Task A와 B가 서로 다른 feature를 만들었을 때, 두 번째 block의 SA는 각 다른 입력을 받게 되고, SA weight도 달라짐

✓따라서 두 task 사이의 차이는 SA 파라미터 안에서도 존재함



Paper 2

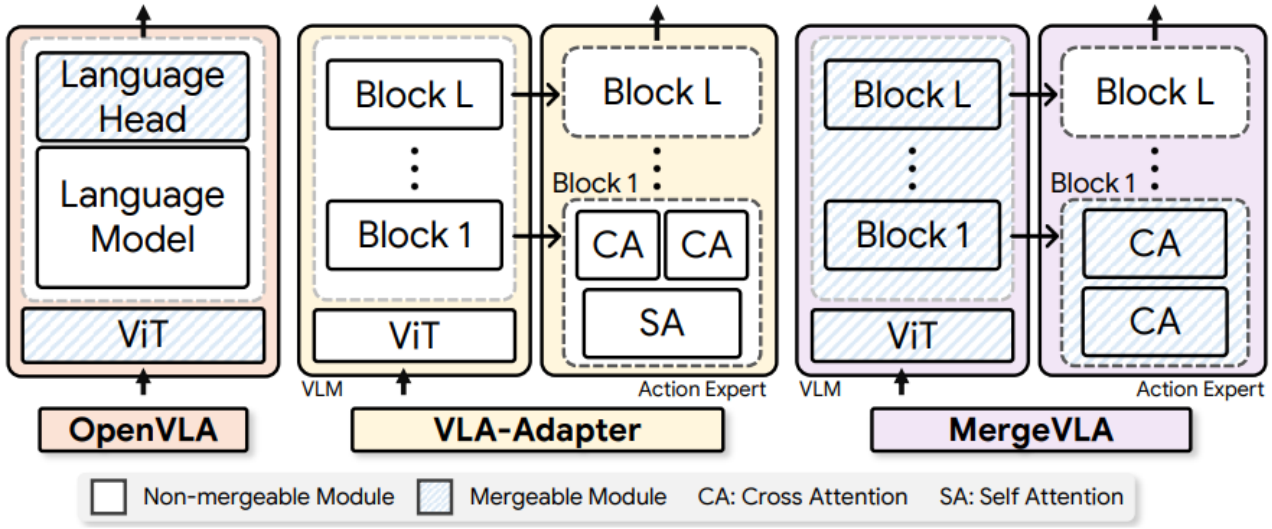


- Motivation 2
  - Self-attention은 이전 block의 action feature를 다시 입력으로 사용함
    - Self-attention이 있는 경우 첫 번째 block에서 생긴 task-specific feature가 다음 block의 self-attention 입력으로 다시 들어감

$\therefore x_{task}^{1} \rightarrow x_{task}^{2} \rightarrow x_{task}^{3} \rightarrow \cdots \rightarrow x_{task}^{L}$

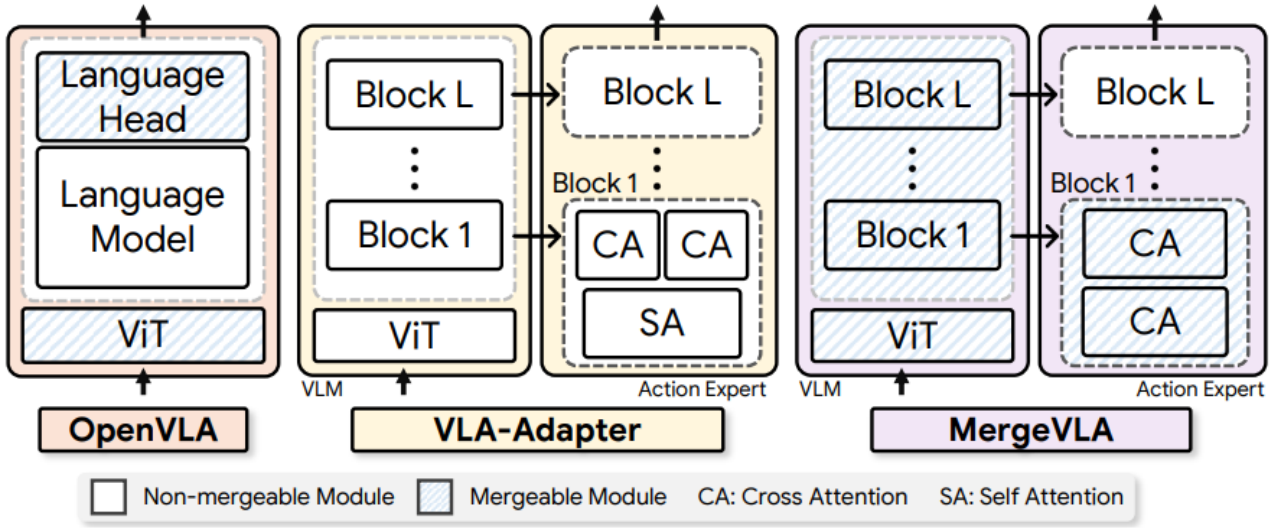
✓이 과정이 block마다 반복되기 때문에, shallow block에서는 task 간 parameter distance가 작지만, 마지막 block 부근에서는 급격히 증가함

✓이를 SA feedback에 의해 task dependency가 깊이에 따라 축적된 결과로 해석함



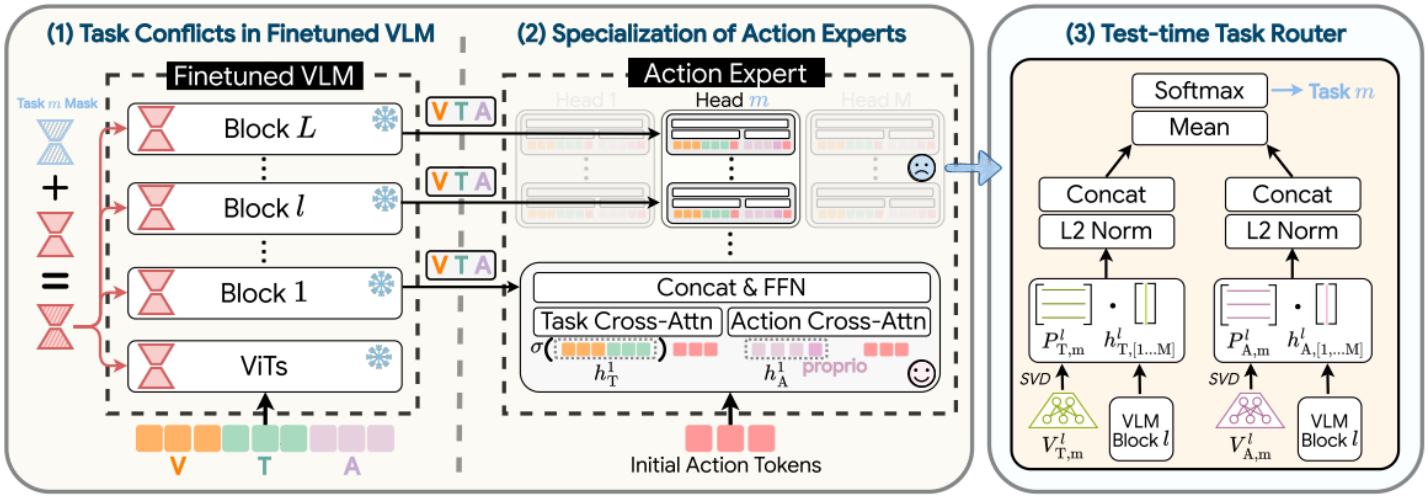
Paper 2

- Motivation 2
  - Cross-attention은 merging이 쉬워짐
    - MergeVLA는 SA를 제거하고 각 block이 VLM feature를 참조하도록 만듦
    - $x^{i+1} = CA(x^i, h_T^i) + CA(x^i, h_A^i) + FFN()$ $\#h_T^i, h_A^i$: VLM이 제공하는 task/action 관련 hidden feature
      - 따라서 각 action block은 이전 block에서 누적된 task-specific action representation보다, 상대적으로 안정적이고 공유 가능한 VLM representation에 계속 anchor 됨
      - 때문에 task specialization이 전체 action expert 깊이에 퍼지지 않고, 마지막 몇 block에 국소화
        - ✓MergeVLA에서는 shallow block들은 평균 병합하고, 마지막 block은 **expert head**로 따로 보존

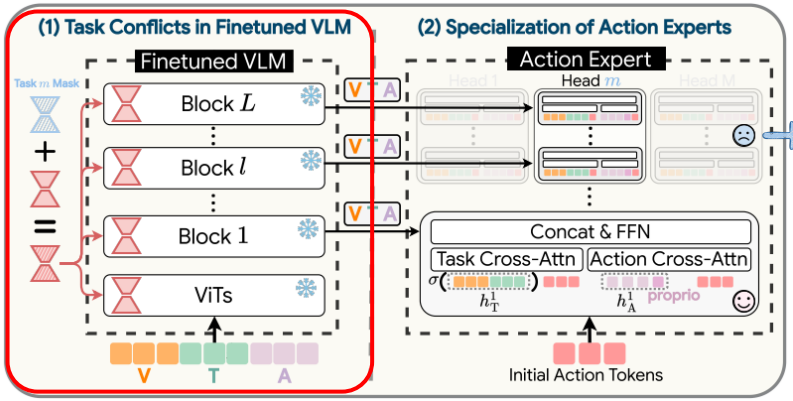


Paper 2

- Method Overview
  - 두 가지 Motivation에 각각 대응하는 설계와 unseen task 상황을 위한 router로 구성된 merge-oriented 아키텍처 제안
    - **Design 1 : Task Masking**: 태스크 마스크로 LoRA를 sparsely 활성화해 VLM merge 안정화 (Motivation 1 대응)
    - **Design 2 : Cross-Attention-Only Action Expert**: SA 제거로 대부분의 block을 병합 가능하게 재설계 (Motivation 2 대응)
    - **Design 3 : Test-time Task Router**: Task identity가 없어도 초기 관측만으로 적절한 mask·expert head를 자동 선택
      - Task identity: 추론 시점에 알려지지 않은 mixed-task



Paper 2



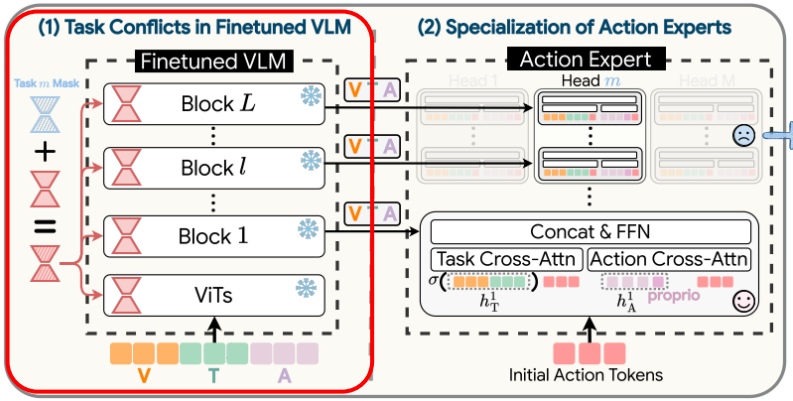
- Method – Task Masking
  - Setup: Task Vector와 기존 병합의 한계
    - 표준 LoRA 업데이트를 하나의 task vector로 정의하고, 이를 병합해 단일 업데이트를 구성하는 것이 기존 data-free 병합의 공식
      - **Task Vector 정의**: $\tau_m = \Theta_m - \Theta_0$
        - ✓Θ_0: 사전학습 checkpoint
        - ✓Θm: task m에 대해 LoRA finetuning된 가중치 (m = 1,…,M)
      - **표준 병합 공식 (TA, TIES, SVD 기반 등)**:

$$\tau_{\text{merge}} = \alpha\,\mathcal{R}(\{\tau_m\}_{m=1}^{M}), \quad \Theta_{\text{merge}} = \Theta_0 + \tau_{\text{merge}}$$

        - ✓(α: scaling factor, R(·): 병합 연산자)
      - **한계**: τ_{merge}는 task 간 충돌로 인해 직접 사용 불가능
        - ✓단일 global update를 넘어서는 접근이 필요

Paper 2

- Method – Task Masking
  - Task-specific Binary Masking
    - Θ_{merge} 내에서 task m에 유익한 성분만 골라내는 binary mask S_m 을 도입
      - 충돌을 인코딩하는 성분은 억제
      - **마스킹된 가중치**: $\Theta_{\text{merge}}^{(m)} = \Theta_0 + \mathbf{S}_m \odot \tau_{\text{merge}}$
        - ✓$\odot$: element-wise multiplication
        - ✓S_m이 1인 위치만 병합된 업데이트를 적용, 나머지는 사전학습 가중치 Θ_0 유지
      - **직관**: task m을 수행할 때는 Θ_{merge}^{m}을 사용
        - ✓m개의 서로 다른 마스킹된 VLM variation이 하나의 병합 파라미터 집합을 공유
      - **다음 질문**: mask S_m은 어떤 기준으로 결정하는가? → parameter-level consistency test

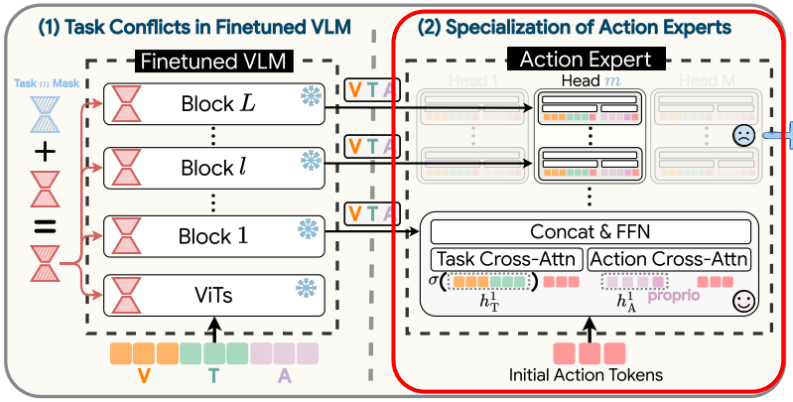


Paper 2

- Method – Task Masking
  - Mask 구성 기준: Parameter-level Consistency Test
    - 파라미터가 task별로 유의미하게 크고 전체 병합 방향과 정렬될 때만 마스크를 1로 설정
      - **마스크 정의**: $\mathbf{S}_m = \mathbb{I}\left[\,|\tau_m| > \lambda\,|\tau_{\text{merge}} - \tau_m|\,\right]$
        - ✓$\mathbb{I}[\cdot]$: 조건이 참이면 1, 거짓이면 0을 반환하는 지시함수
        - ✓λ (mask ratio) : 불일치 허용 정도를 조절하는 hyperparameter
      - **해석**:
        $|\tau_m|$이 residual $|\tau_{merge} - \tau_m|$보다 충분히 크다 →
        이 파라미터는 task m 고유의 뚜렷한 신호이자 전체 병합 방향에도 부합 →
        신뢰하고 유지

Paper 2

- Method – Action Expert
  - Action Expert 재설계: 대상 아키텍처 분석
    - VLM을 완벽히 병합해도 VLA-Adapter의 action expert를 단순 평균화하면 성공률 0%
      - masking만으로는 불충분
      - **구성 (처음부터 학습, block L개)**:
        ① Self-Attention → ② Task Cross-Attn(tanh gate) → ③ Action Cross-Attn → ④ FFN
      - ①이 task-specific bias를 depth에 따라 누적·증폭 (Motivation 2)하여 문제 발생

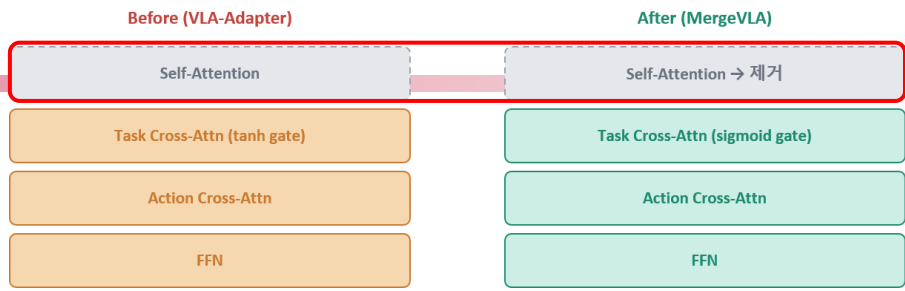


Before (VLA-Adapter)

- Self-Attention
- Task Cross-Attn (tanh gate)
- Action Cross-Attn
- FFN

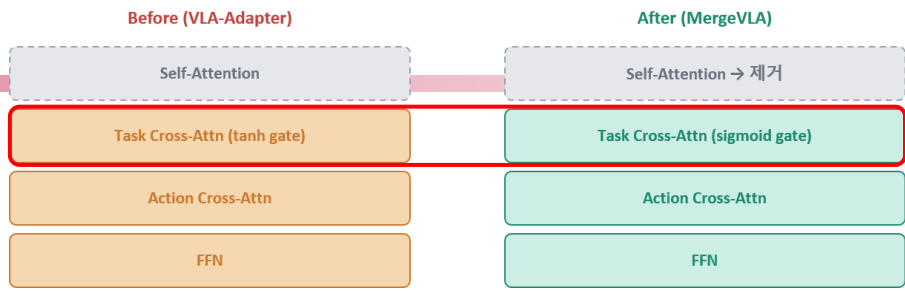
After (MergeVLA)

- Self-Attention → 제거
- Task Cross-Attn (sigmoid gate)
- Action Cross-Attn
- FFN



Paper 2

- Method – Action Expert
  - 수정 1: Self-Attention 제거
    - Action expert의 self-attention layer를 완전히 삭제하고 cross-attention만 남김
      - 남은 유일한 정보 경로는 VLM에서 오는 hidden state
    - **왜 self-attention이 문제인가**:
      Action expert 전체가 로봇 시연 데이터로 처음부터(from-scratch) 학습됨
      - Self-attention은 block 입력 x^i에 대해 직접 작동
        - ✓학습 과정에서 task별 고유 패턴을 자유롭게 형성하고 이를 깊이에 따라 전파
      - 모듈적 변환을 제공하는 대신, task-dependent 신호를 증폭
        - ✓깊은 block이 개별 task에 병리적으로 특화됨
    - **제거 후 효과**: Self-attention 경로가 사라지면서 expert는 오직 VLM이 제공하는 공유된 feature(h_T, h_A)에만 의존하도록 강제됨
    - **결과**: task-specific bias의 원천이 제거되어, 대부분의 block이 단순 가중치 평균화로도 병합 가능해짐

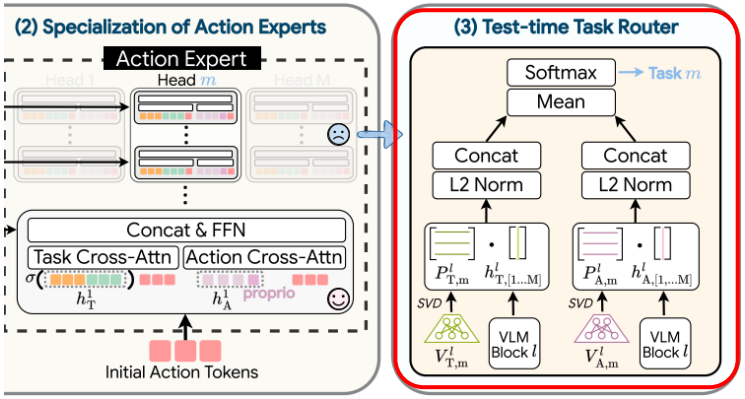


Paper 2

- Method – Action Expert
  - 수정 2: Gating Function 교체 (tanh → sigmoid)
    - Task cross-attention 경로에 적용되는 gate를 tanh에서 sigmoid로 교체
      - VLM 정보가 항상 균형 있게 보존되도록 함
    - **기존 gate (VLA-Adapter)**:
      - 의 출력 범위는 (−1, 1) : 음의 활성화가 가능해, VLM이 전달하는 신호를 능동적으로 억제(상쇄)할 수 있음
      - 이 경우 expert는 VLM 신호 대신 자신의 scratch-trained(=task-specific) 파라미터에 더 의존하게 되는 부작용 발생
    - **변경된 gate (MergeVLA)**: , $g(\cdot) = \text{sigmoid}$, 출력 범위 (0, 1)
      - 음의 활성화가 불가능 →
      - VLM이 제공하는 정보가 항상 **보존 또는 축소**만 되고 **반전**되지 않음 →
      - 공유 feature에 대한 의존도가 안정적으로 유지
    - **두 수정의 결합 효과**: LIBERO-Plus(OOD)에서 VLA-Adapter 대비 성공률 +13.4%
    - **처리 방식**: Expert head는 병합하지 않고 task별로 그대로 유지하며 나머지 block만 병합해 재사용

Paper 2

- Method – Test-time Task Router
  - 문제 설정: Task identity가 없을 때
    - Task identity를 알고 있다면 해당 mask S_m과 expert head $H_{(l \to L)_m}$을 수동 선택하면 되지만, mixed-task 평가에서는 이것이 불가능함
      - **기존 접근의 한계**: 많은 방법이 task identity나 task별 prompt 같은 수작업 사전 정보(hand-picked prior)에 의존
        - ✓이것이 없으면 성능이 급격히 저하됨
      - **MergeVLA의 목표**: 추가 학습이나 supervision 없이, 오직 입력 observation만으로 적절한 mask·head를 동적으로 선택
      - **Test-time Task Router 목표:** 모델의 내부 파라미터 subspace로부터 직접 task 관련성을 추론하는 task routing 메커니즘 설계



Paper 2

- Method – Test-time Task Router
  - Router 알고리즘 (Step 1-2): Masked VLM 구성 & SVD
    - 각 후보 task마다 서로 다른 mask를 적용한 **m개의 masked VLM**을 만들고, action expert의 value projection을 SVD로 분해해 대표 subspace를 추출

Step 1. Masked VLM 구성:

✓$\Theta^{m}_{merge} = \Theta_0 + S_m \odot \tau_{merge} \ (m = 1, \dots, M)$

✓각 masked VLM을 초기 관측(I_0^v, I_0^w, L)에 대해 실행

✓block l에서 hidden state $\{h_T^l, h_A^l\}$ 추출

Step 2. Value Projection SVD 분해:

✓$V_T^l = L_T^l \Sigma_T^l (R_T^l)^{\mathrm{T}}, \quad V_A^l = L_A^l \Sigma_A^l (R_A^l)^{\mathrm{T}}$

✓각 경로에서 상위 k_r개 right singular vector를 취해 dominant subspace $P_T^l, P_A^l \in R^{k_r \times d}$구성

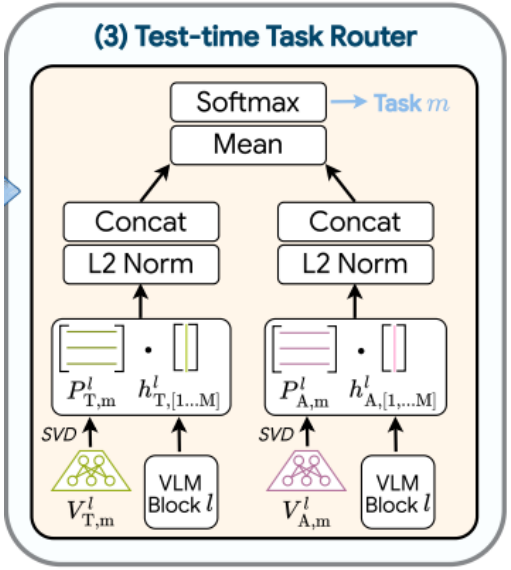


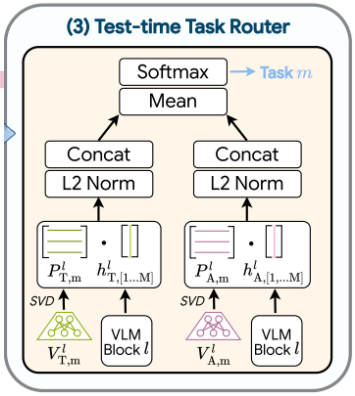
Table 5. Ablation results of MergeVLA with different subspaces used for routing on LIBERO.

Method	Spatial	Object	Goal	Long	Avg.
Only K	**98.0**	0.0	39.6	**76.6**	53.6
K&V	**98.0**	0.0	**85.8**	**76.6**	65.1
Only V	**98.0**	**98.8**	85.4	**76.6**	**89.7**

Paper 2

- Method – Test-time Task Router
  - 설계 근거: 왜 Query/Key가 아닌 Value인가
    - Ablation 결과 Value 기반 subspace가 Query/Key 기반보다 훨씬 안정적
      - 이는 attention 메커니즘의 역할 차이에서 비롯됨
      - **Query/Key의 역할**: attention의 similarity 구조를 정의
        - ✓'어디를 볼 것인가'를 결정하는 역할
        - ✓입력 스케일링에 민감하고, task-specific subspace로 쉽게 붕괴하는 경향이 있어 router 신호로는 불안정
      - **Value의 역할**: attention이 실제로 retrieve하는 내용, 즉 hidden state에 기록되는 행동적 의미(task 정보)를 직접 인코딩
        - ✓더 안정적이고 판별력 있는 task 식별 신호 제공
      - **실증 결과 (LIBERO, 4개 suite 평균)**: Only-K 53.6% < K&V 65.1% < Only-V 89.7%
        - ✓Object·Goal task에서 K 기반은 router가 task를 잘못 배정하는 오류가 두드러짐
        - ✓Value 기반만이 4개 suite 모두에서 안정적

Paper 2



- Method – Test-time Task Router
  - Router 알고리즘 (Step 3-4): 점수화 & Task 선택
    - 추출한 subspace에 hidden state를 투영해 활성화 강도를 측정하고, 가장 높은 관련도를 보이는 task를 선택해 episode 내내 고정

Step 3. 관련도 점수 계산:

✓$r_{T,m} = \left\| P_T^l \cdot h_A^l \right\|_2, \quad r_{A,m} = \left\| P_A^l \cdot h_T^l \right\|_2$

✓결합 점수: $r_m = \frac{1}{2}\left(r_{T,m} + r_{A,m}\right)$→ 모든 후보 m에 대해 $r \in R^M$ 벡터 구성

Step 4. Task 선택 & 고정:

✓$p = softmax(r), \quad m* = argmax_m p_m$

✓t=0의 초기 관측 1회만으로 충분

✓이후 episode 동안 $S_{(m*)}$와 $H_{(l \to L)m*}$를 고정 사용, 반복 라우팅 불필요

Overhead: m개의 task mask와 head를 유지해야 하지만, 추가 연산·파라미터 부담은 미미함

Paper 2

- Experiments
  - LIBERO / LIBERO-Plus / RoboTwin 2.0 / Real-world SOARM-101
    - 4개 벤치마크로 mixed-task·OOD·cross-embodiment 성능을 검증

    - **LIBERO**: 4개 suite × 10 task, task당 50 시연 (in-distribution 병합 평가)
    - **LIBERO-Plus**: 10,030개 task, 7종 perturbation (OOD 견고성 평가)
    - **RoboTwin 2.0**: dual-arm 로봇 3종 × 4개 task (cross-embodiment 평가)
    - **실제 SO-101 로봇**: 3개 조작 task, task당 시연 50개 (Real-world 배치 검증)
    - **구현**: backbone Qwen2.5-0.5B, single A6000 GPU

Paper 2

- Results
  - MergeVLA+TIES는 LIBERO mixed-task에서 90.2% 달성
    - Single-task finetuning(96.7%) 대비 6.5% 차이로 기존 병합 기법(TA=0%) 개선
    - TA(**All**) 0% → **OpenVLA**+TA+Mask 62.4% → **VLA-Adapter**+Mask 23.1% → **MergeVLA**+TIES 90.2%

Method	Merge Method	Merge Part	Params (B)	Spatial	Object	Goal	Long	Avg.
Single-task Finetuned Model								
OpenVLA [27]	-	-	7 × 4	84.7	88.4	79.2	53.7	76.5
VLA-Adapter [45]	-	-	0.68 × 4	99.6	99.6	98.2	96.4	98.5
MergeVLA	-	-	0.68 × 4	98.0	98.6	95.0	95.0	96.7
Merged Model								
OpenVLA	TA [21]	Vision Backbones	7 × 4	56.6	58.0	55.6	6.6	44.2
OpenVLA	TA [21]	All	7	0.0	0.0	0.0	0.0	0.0
OpenVLA	TA [21] + **S**	All	7	74.2	82.6	68.8	24.0	62.4
VLA-Adapter	TA [21]	All	0.68	0.0	0.0	0.0	0.0	0.0
VLA-Adapter	TA [21] + **S**	All	0.68	0.0	0.0	0.0	0.0	0.0
VLA-Adapter	TA [21] + **S**	Except $\mathbf{H}^{L\to L}$	0.70	50.2	34.6	0.0	7.4	23.1
$\text{MergeVLA}_{\text{EMR}}$	EMR [20]			96.0	63.2	62.0	40.6	65.5
$\text{MergeVLA}_{\text{TSV}}$	TSV [17] + **S**			**99.4**	97.8	74.4	54.8	81.6
$\text{MergeVLA}_{\text{KnOTS}}$	KnOTS[39] + **S**			96.8	98.8	84.8	71.4	88.0
$\text{MergeVLA}_{\text{TA}}$	TA [21] + **S**	Except $\mathbf{H}^{L\to L}$	0.70	98.0	**98.8**	85.4	76.6	89.7
$\text{MergeVLA}_{\text{WUDI}}$	WUDI [8] + **S**			97.6	98.2	85.6	78.2	89.9
$\text{MergeVLA}_{\text{TIES}}$	TIES [48] + **S**			94.8	94.6	**91.8**	**79.4**	**90.2**

Paper 2

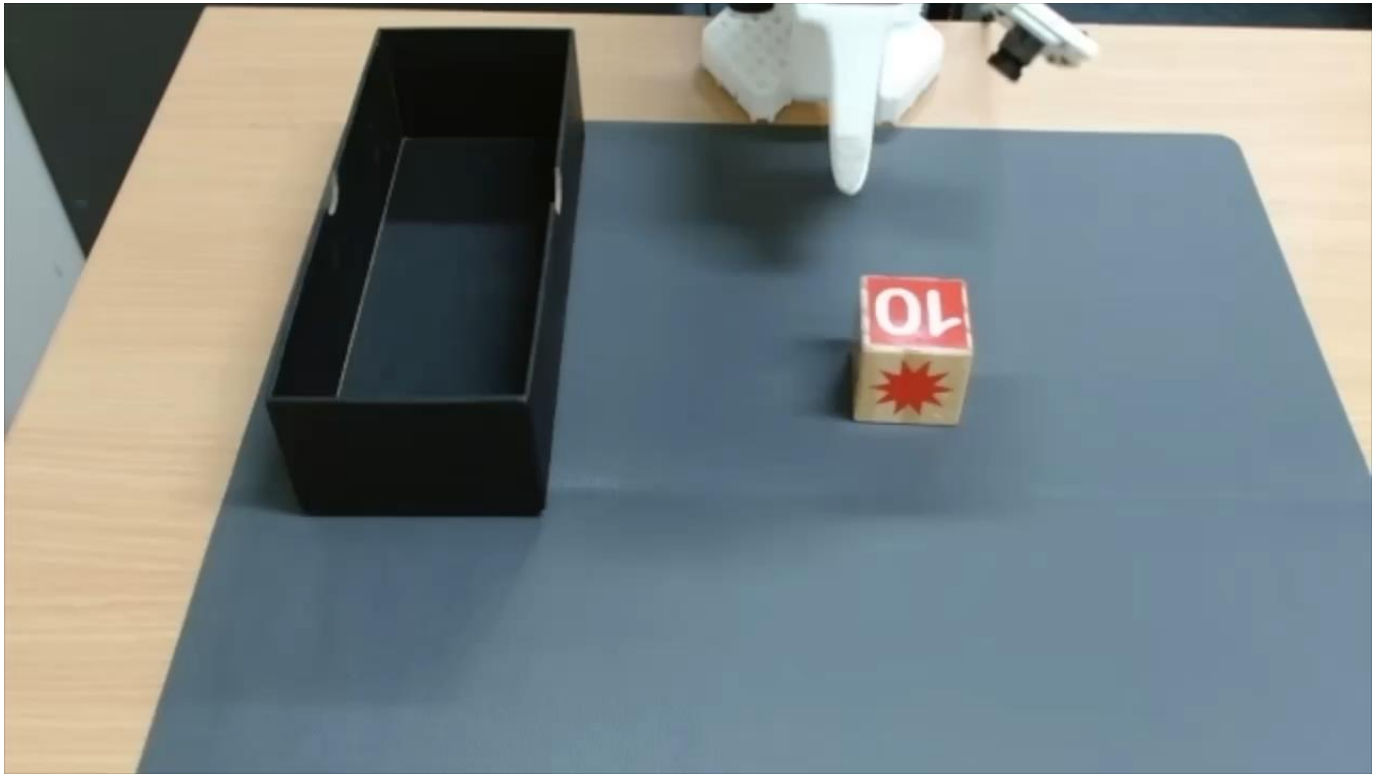
- Results
  - LIBERO-Plus(OOD): 62.5%
    - Single-task finetuning model에서도 기존 SOTA보다 더 견고

Method	S1	S2	S3	S4	S5	S6	S7	Avg.
Single-task Finetuned Model								
OpenVLA [27]	34.8	0.8	23.0	8.1	28.5	3.5	15.2	16.3
π_0 [3]	81.4	13.8	58.8	85.0	68.9	6.9	**79.0**	56.3
VLA-Adapter [45]	76.6	36.4	73.8	71.0	70.2	37.4	57.2	59.0
MergeVLA	92.7	**62.4**	75.7	**92.7**	73.7	**46.4**	74.7	**72.4**
Merged Model								
VLA-Adapter [45]	15.7	6.6	17.6	11.2	15.0	4.1	7.1	10.8
MergeVLA$_{\text{TSV}}$	64.3	44.8	58.8	72.9	59.4	30.4	52.7	53.5
MergeVLA$_{\text{TA}}$	78.2	**53.1**	**68.4**	79.0	65.0	**34.6**	62.7	61.6
MergeVLA$_{\text{TIES}}$	**85.7**	50.7	66.0	**84.2**	**68.1**	30.3	**66.0**	**62.5**

Paper 2

- Results
  - **Real-world (SOARM-101)**

Method	Pick & Place	Push Cube	Stack Cube	Avg.
Single-task finetune	90.0	85.0	95.0	90.0
$\text{MergeVLA}_{\text{TA}}$	70.0	70.0	60.0	66.7
$\text{MergeVLA}_{\text{TIES}}$	**90.0**	**90.0**	**90.0**	**90.0**



Thank You