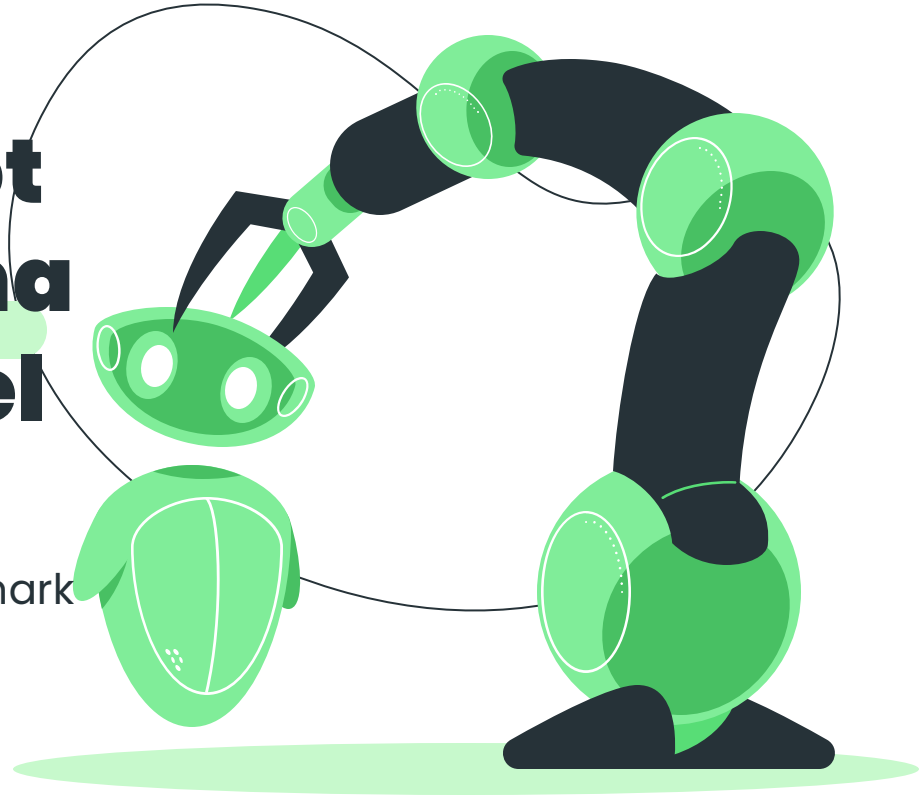
Multimodal Robot Control via Magma Foundation Model

Yuhe Bian, Biyu Zou
Supervision: Hashim Ismail, Maj Stenmark



What is Magma?

Multimodal Agentic Intelligence

- Perceive images and sensor data
- Plan actions
- Act

Spatial-Temporal Intelligence

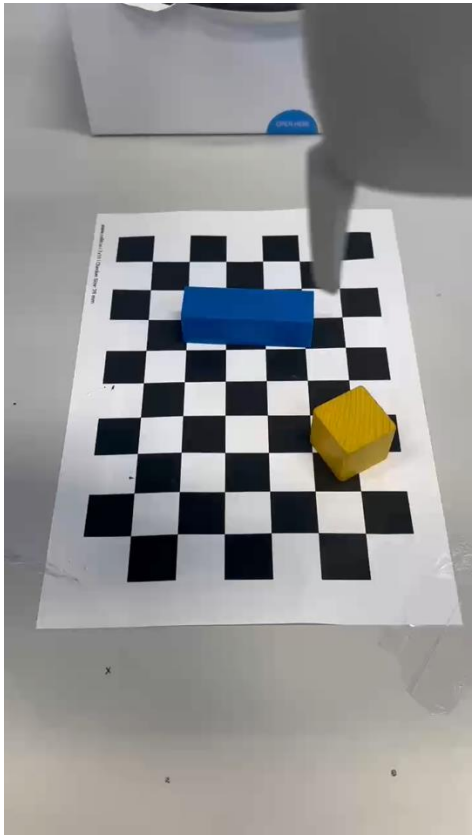
- Spatial reasoning
- Temporal dynamics

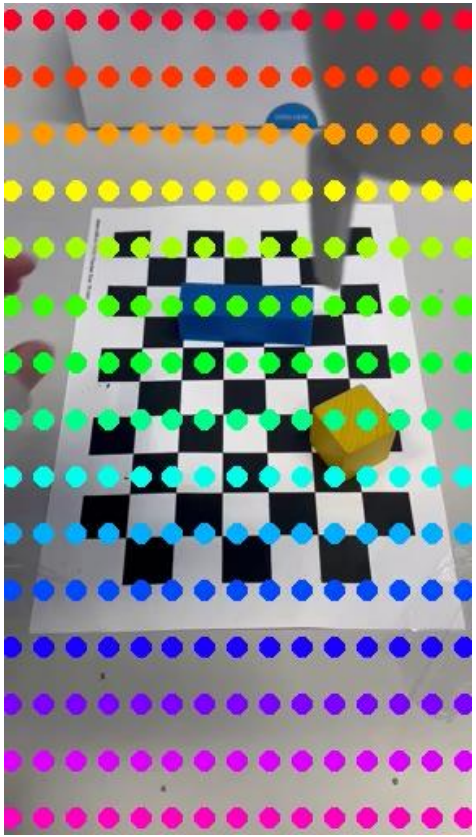
Set of Marks & Trace of Marks

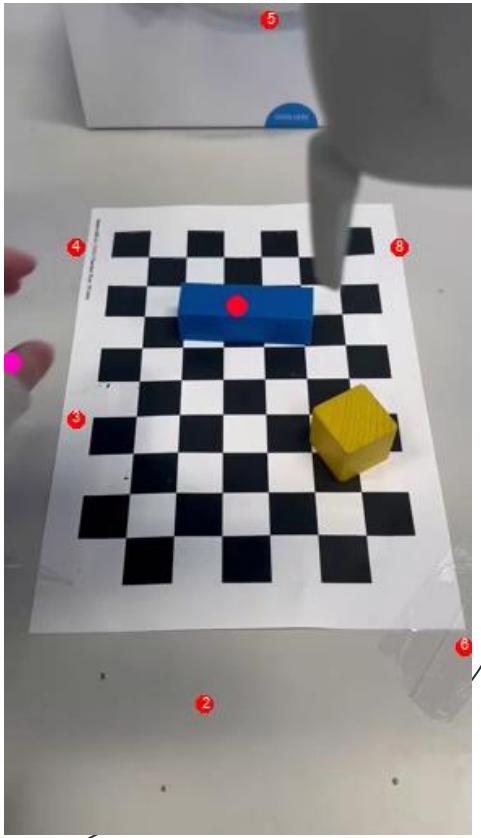
SoM: marks for interactive targets

ToM: Annotated trajectories of hands or objects

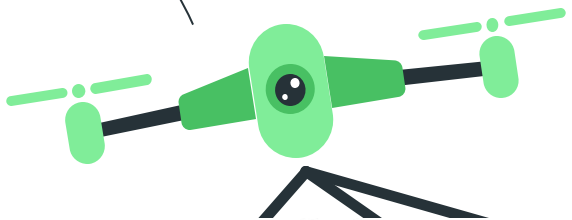
Set of Marks & Trace of Marks

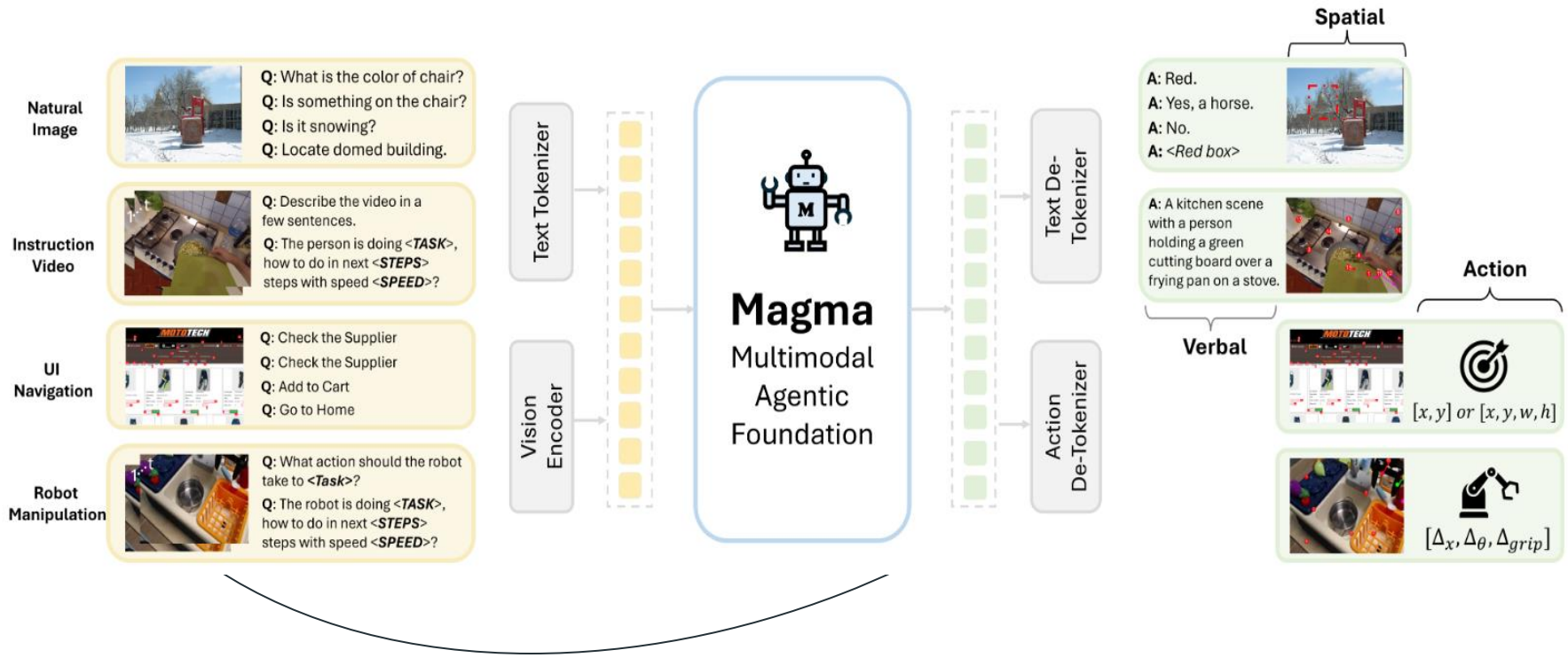






Magma Structure





Original Goal

1

Control Robot Arm with ROS

2

Finetune Model with Arm Data

3

Use Model to Predict and Execute Actions

Key Challenges

Model Architecture Limitations

-Language-only model with no native action tokens

-Insufficient special token space (<256), limits extensibility

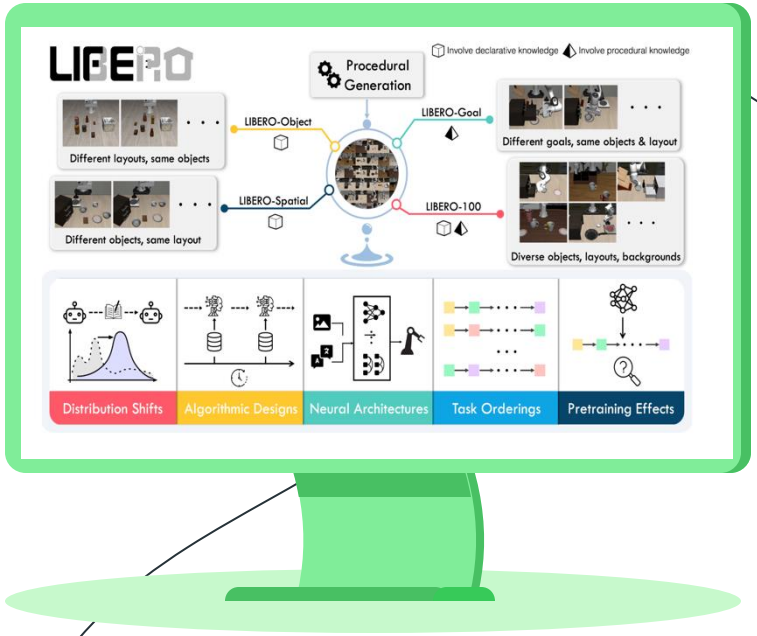
Deployment Constraints

-No released robot-specific checkpoint

-Poor generalization to real-world, unseen environments

Why We Use the LIBERO Dataset

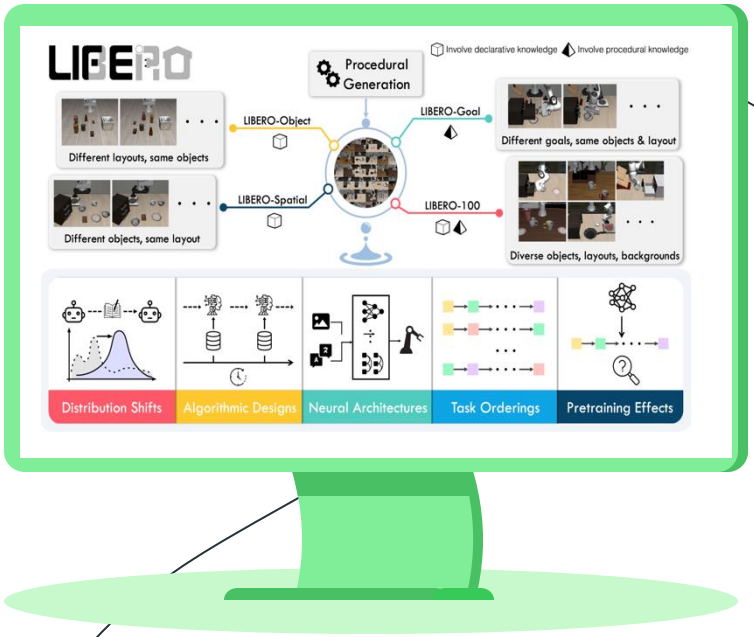
- Rich task diversity
- Supports Efficient Few-shot Fine Tuning
- Well-annotated trajectories
- Aligned with Magma Format



Dataset overview

- **Libero-Spatial** subset
- **432** trajectories, total **52,970** steps (transitions)
- Actions are **pre-discretized** into bins before training

(image at this step + task instruction) ⇒ discrete action token



Encoder

continuous 7D actions
(Δx ,Δy ..$\Delta\theta$,gripper)

256 bins

-1 ● ● ● ● ● ● ● ● ● 1

digitization

<action_*>
discrete action token

token_id

embedding

→

Embedding Table → E

Hidden State → h

logits = $h^T E$

Softmax Predicted Probabilities

→

Decoder

output token ID

bin index

bin center value

final continuous action

Model Structure

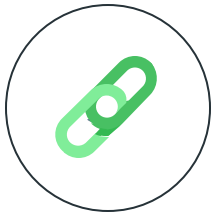
Technical Details

LoRA Fine-tuning

Fine-tuning all the linear layers with rank=32

PEFT Integration

Use HuggingFace's PEFT framework to plug in and manage LoRA modules efficiently.



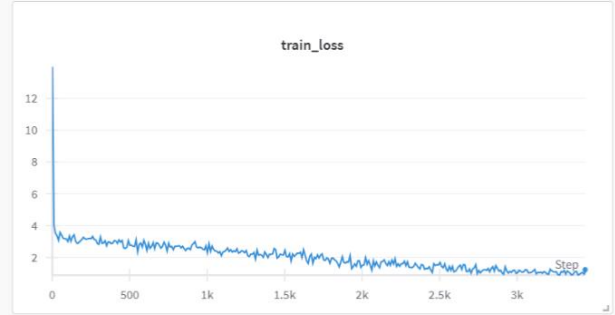
Action Token Mapping

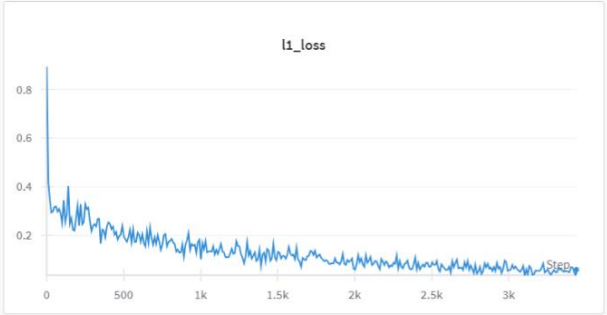
Maps 256 discretized actions into the least-used vocabulary slots.

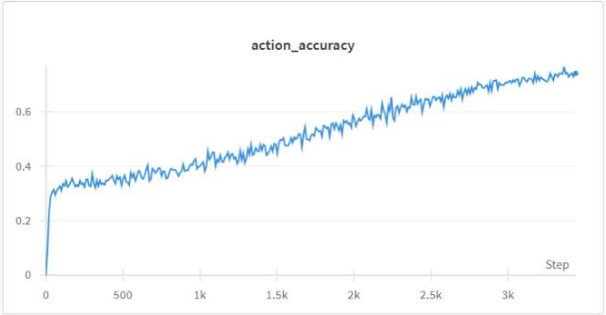
Distributed Training (DDP)

Leverages PyTorch DDP for multi-GPU parallel fine-tuning.

Training







- Batch Size: 16 (with gradient accumulation = 1)
- Max Steps: 200,000
- Learning Rate: 5e-4
- Checkpointing: Saved every 5,000 steps
- 4-bit Quantization
- LoRA Rank = 32
- LoRA Dropout = 0.0

Future Works

Evaluation	Using LIBERO-Goal — diverse robot manipulation tasks
Real-world deployment	Connect the model directly to the UR5e robot

Thank you!