ELV26 Tutorial 2: **Embodied Simulators**

Ellis Brown

2026-02-03

About Me

- 3rd year PhD w/ Profs. Saining Xie & Rob Fergus

- Focus:
  - **Multimodal LLMs:** single image → multi-image / video
  - **Spatial Understanding / EAI**: visuospatial reasoning, learning from simulators

- Select Papers:
  - Cambrian-1: vision-centric MLLMs
  - "Test-set Training": identifying & mitigating non-visual shortcuts in benchmarks
  - SIMS-V: learn spatial video-understanding via simulators
  - Cambrian-S: spatial *supersensing* in video

Open OnDemand Setup

https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/greene/software/open-ondemand-ood-with-condasingularity

1. Create Conda environment with Singularity *on* Burst's /scratch
   a. https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/greene/software/singularity-with-miniconda
2. Setup JupyterLab to work with Singularity+Conda
3. Work directly with JupyterLab

Coding options

1. Use vim directly in terminal on Burst's /scratch
2. Use code server via OOD
3. Do most of coding on local VSCode and test on Burst's compute nodes
4. Setup VSCode to do proxy jumps to Burst's compute nodes (need to always switch ssh config)

Limitations of *static* computer vision

***Static* computer vision:** learning from still images or videos and performing tasks like object detection, image classifcation, scene recognition

***Embodied* visual intelligence:** learning by interacting and modifying the environment. Learning *passively* targeting an embodied POV

How can we achieve this type of learning and visual intelligence?

Robots?

Robots?

Why not?

- **Scalability:** Physical collection is slow (RT-1 took 17 months)
- **Cost & Safety:** Can't crash a real car 1,000 times to learn safety
- **Reset-ability:** Hard to automatically "reset" a messy room in the real world
- **Hardware Transfer:** Policies struggle to transfer between different robot kinematics

Embodied simulators

Software that allows us to render and interact with realistic environments

Benefits

1. **Speed:** run many simulations at once at high speeds
2. **Reproducibility:** control initializations, conditions, etc.
3. **Low-cost:** test expensive and dangerous settings freely
4. **Scalable:** generate large volumes of precise data

Embodied simulators

Primary drawback is the ***sim-to-real gap***: training on simulation may not transfer to the real world

1. Object appearance, properties
2. Physics, motion
3. Limited interactions, actions

Simulator components

Physics engine: models changes in world state over time

- PyBullet, MuJoCo, DART, ODE, PhysX, FleX, Chrono

Renderer: generate observations from states

- Magnum, ORRB, PyRender

Examples that do both: Unity, Unreal

Simulator Landscape

Indoor Navigation & Interaction

- AI Habitat
- AI2-THOR
- ThreeDWorld
- iGibson

Robotics & Manipulation

- NVIDIA Isaac Sim
- MuJoCo
- Genesis
- RLBench
- SAPIEN

Autonomous Driving

- nuPlan
- CARLA Simulator
- Waymo Open

Humanoid

- NVIDIA Isaac Lab
- Habitat 3.0

→ World Models as “simulators”?

- **Classic Simulators** (Habitat/Isaac): "Engine-based." Physics are hard-coded (Newtonian). Ground truth is ***perfect***. Used for training agents.
  ⇒ ***Correctness***

- **World Models**: "Learned." Physics are predicted by a neural net. Used for planning or imagination.
  ⇒ ***generalization?***
  - Physical AI with World Foundation Models | NVIDIA Cosmos
  - Dreamer 4: Training Agents Inside of Scalable World Models
  - Genie 3 — Google DeepMind
  - World Labs
  - Sora | OpenAI

Indoor simulators

	Rendering		Physics		Scene	Speed
	Library	Supports	Library	Supports	Complexity	(steps/sec)
Habitat [3]	Magnum	3D scans	none	continuous navigation (navmesh)	building-scale	3,000
AI2-THOR [6]	Unity	Unity	Unity	rigid dynamics, animated interactions	room-scale	30 - 60
ManipulaTHOR [26]	Unity	Unity	Unity	AI2-THOR + manipulation	room-scale	30 - 40
ThreeDWorld [7]	Unity	Unity	Unity (PhysX) + FLEX	rigid + particle dynamics	room/house-scale	5 - 168
SAPIEN [34]	OpenGL/OptiX	configurable	PhysX	rigid/articulated dynamics	object-level	200 - 400†
RLBench [35]	CoppeliaSim (OpenGL)	Gouraud shading	CoppeliaSim (Bullet/ODE)	rigid/articulated dynamics	table-top	1 - 60†
iGibson [36]	PyRender	PBR shading	PyBullet	rigid/articulated dynamics	house-scale	100
Habitat 2.0 (H2.0)	Magnum	3D scans + PBR shading	Bullet	rigid/articulated dynamics + navmesh	house-scale	1,400

Considerations

1. Dataset size, scene size
2. Realism of physics and scenes, sim2real transfer
3. Supported agent action set
4. Simulator speed
5. Established benchmark tasks

AI2-THOR

iTHOR RoboTHOR [1] ProcTHOR-10K [2] ArchitecTHOR [2]

Environment:

1. iTHOR: room-sized 3D scenes
2. RoboTHOR: maze-styled dorm-sized 3D scenes for sim-to-real
3. ProcTHOR: Large diverse house-sized 3D scenes
4. ArchitecTHOR: evaluation larger house-sized 3D scenes

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-Thor: An interactive 3D environment for visual AI. arXiv preprint arXiv:1712.05474, 2017.

AI2-THOR

ManipulaTHOR [5] StretchRE1 [14] LoCoBot [24] Abstract Drone [42]

Agents:

1. ManiulaTHOR, StretchRE1: arms to grasp and open objects
2. LoCoBot, Abstract, Drone: high-level commands like OPEN, PICKUP

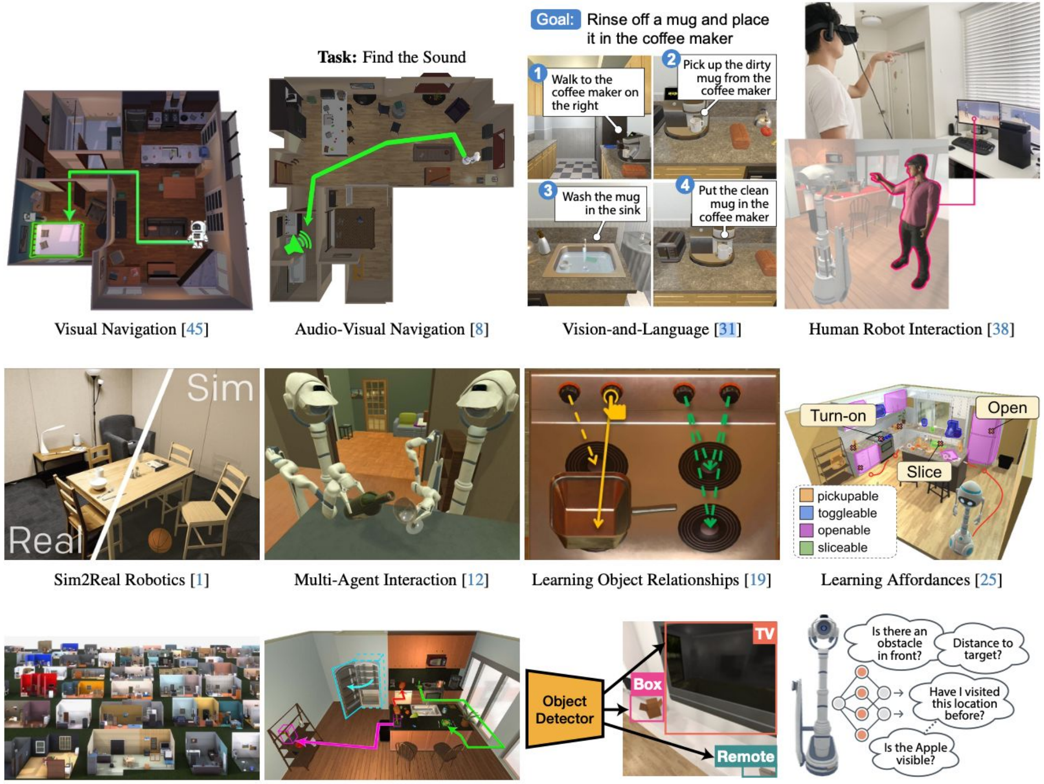
AI2-THOR

Tasks:

1. (Audio)-Visual navigation
2. Vision-language instruction following, question-answering
3. Human-robot interaction
4. Sim2real transfer
5. Multi-agent interaction
6. Object relationships
7. Object affordances

(a) Navigating (b) Changing Object States (c) Opening an Object (d) Grasping an Object (e) Finding the Shortest Path (f) Randomizing Materials

AI2-THOR



Visual Navigation [45] | Audio-Visual Navigation [8] | Vision-and-Language [31] | Human Robot Interaction [38]

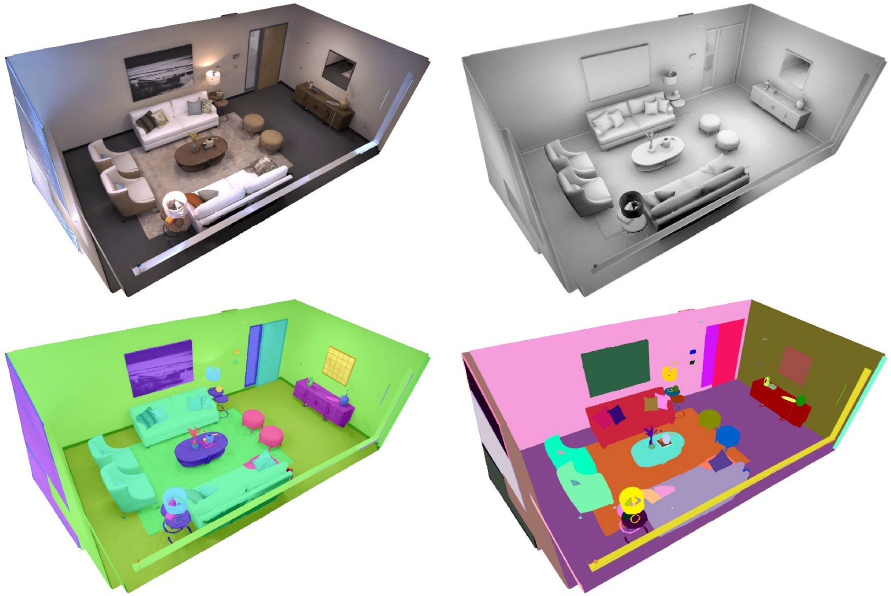
Sim2Real Robotics [1] | Multi-Agent Interaction [12] | Learning Object Relationships [19] | Learning Affordances [25]

Scene Synthesis [2] | Learning with Interaction [34] | Computer Vision [17] | Interpretability [4]

Habitat 2.0

Environment: ReplicaCAD

1. Replica: photo-realistic 3D reconstructions of rooms and buildings
2. Recreate via 3D modeling to make objects interactive
3. Fast: localized physics and rendering



Agent: Fetch

1. Wheeled base, 7-DoF arm
2. 2 RGBD cameras
3. GPS + Compass

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In NeurIPS, 2021

Habitat 2.0 Tasks

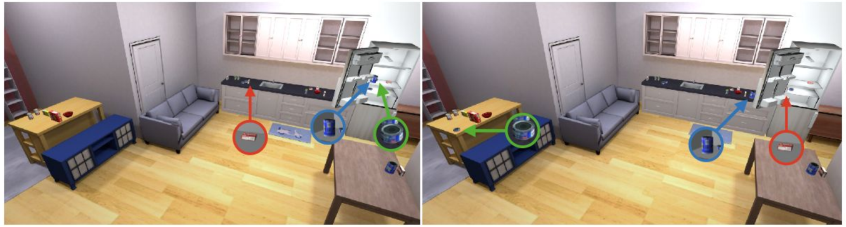
Base: pick task – pick up object from a receptacle

Home Assistant Benchmark:

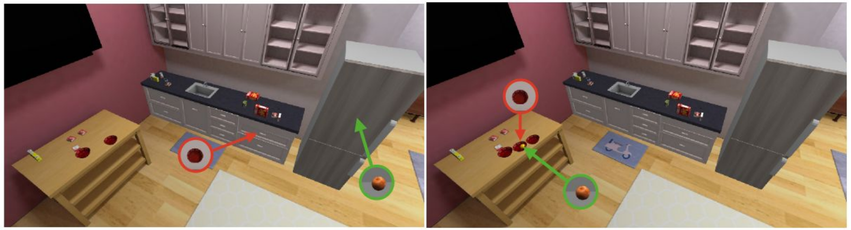
1. TidyHouse
2. PrepareGroceries
3. SetTable

Challenge: long-horizon planning

(a) TidyHouse

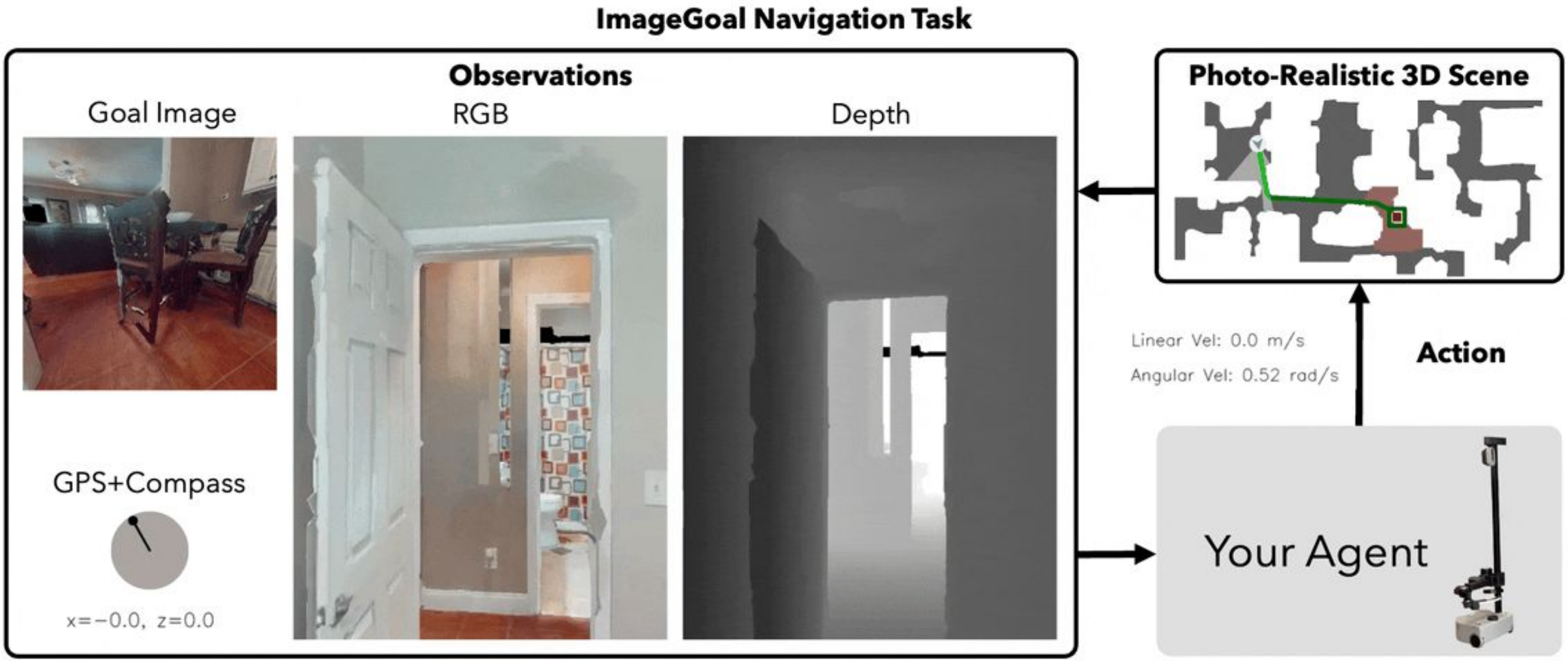


(b) PrepareGroceries

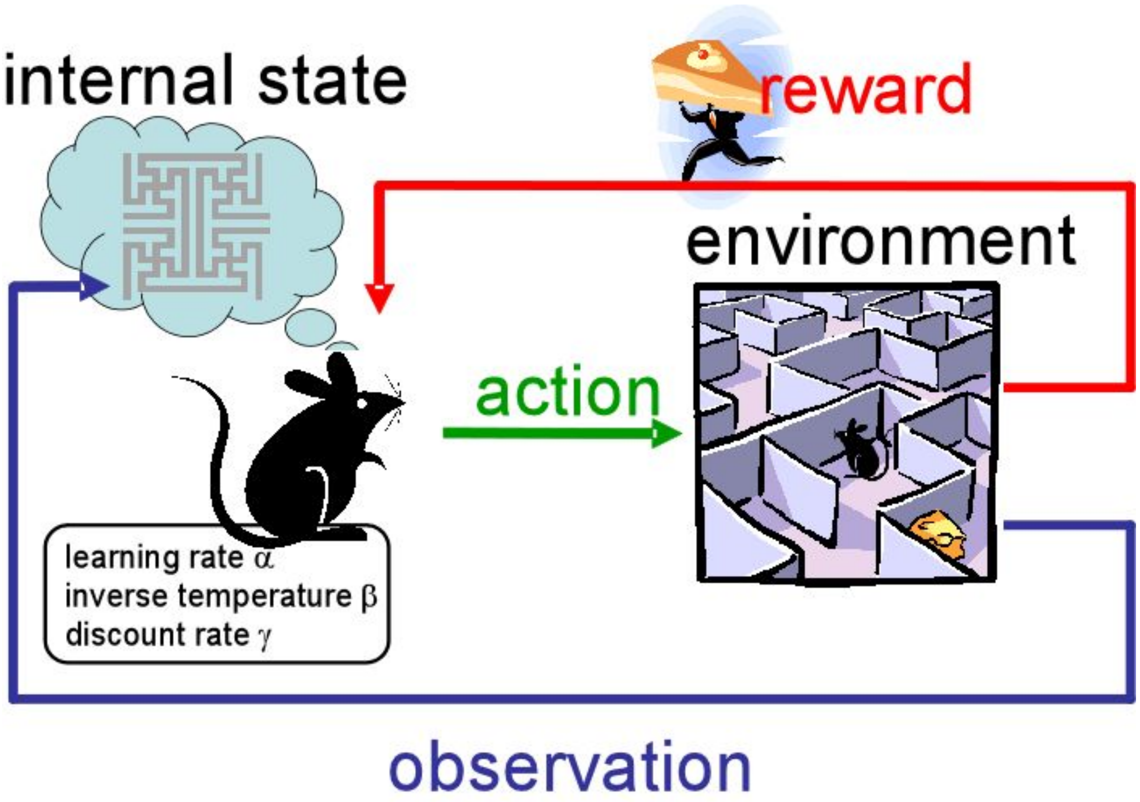


(c) SetTable

Habitat Demo: Image Navigation



Habitat Demo: Reinforcement Learning



Habitat Demo: Reinforcement Learning

We want to learn some policy…

$$a_t = \mu_\theta(s_t)$$
$$a_t \sim \pi_\theta(\cdot|s_t).$$

that maximizes the return, i.e. discounted, cumulative reward

$$r_t = R(s_t, a_t, s_{t+1})$$

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

The overall RL problem then is

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t).$$

$$J(\pi) = \int_\tau P(\tau|\pi) R(\tau) = \mathop{\mathrm{E}}_{\tau\sim\pi}[R(\tau)].$$

$$\pi^* = \arg\max_\pi J(\pi),$$

Habitat Demo: Deep Q Learning

DQN algorithm: goal is to learn the Q-function

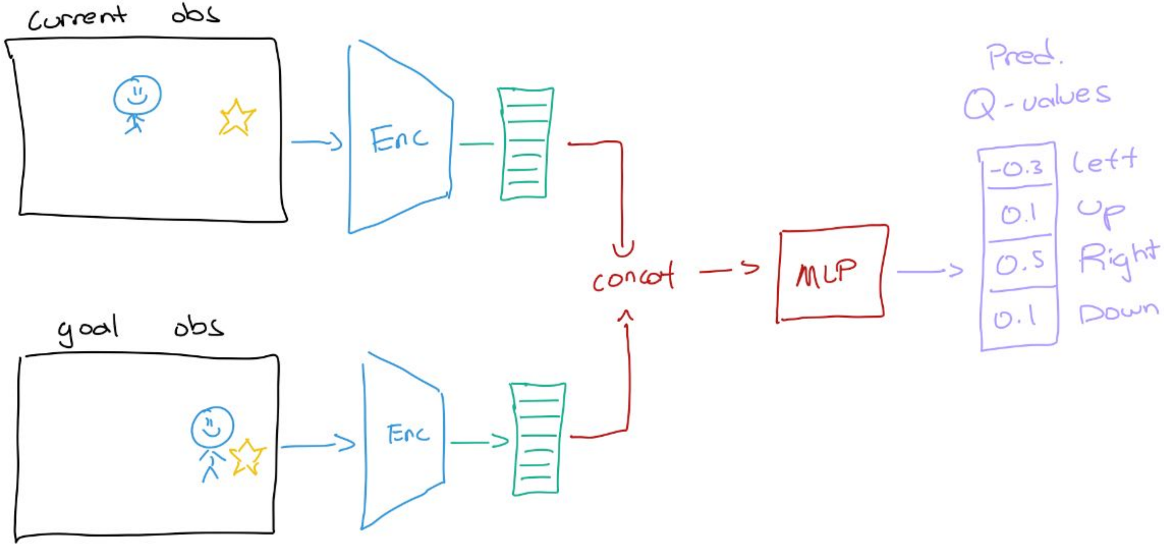
$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \,|\, s_t = s,\ a_t = a,\ \pi\right],$$

$$L_i(\theta_i) = \underbrace{\mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)}}_{\text{Samples from experience replay buffer}} \left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta_i^-)}_{\text{Target network reward estimate at t+1}} - \underbrace{Q(s, a; \theta_i)}_{\text{Q-network reward estimate}} \right)^2\right]$$

Details:

- Samples are from a replay buffer containing *offline* trajectories, i.e. those collected using outdated policies
- Can use epsilon-greedy policy to introduce noise for exploration
- Only works with discrete action spaces

Habitat Demo: Model Architecture



Habitat Demo

See instructions here

https://github.com/embodied-learning-vision-course/course-public/tree/main/2026-spring/labs/lab2