SMILAB

2023-2024

ADVISOR

SUGIURA Komei

STUDENT

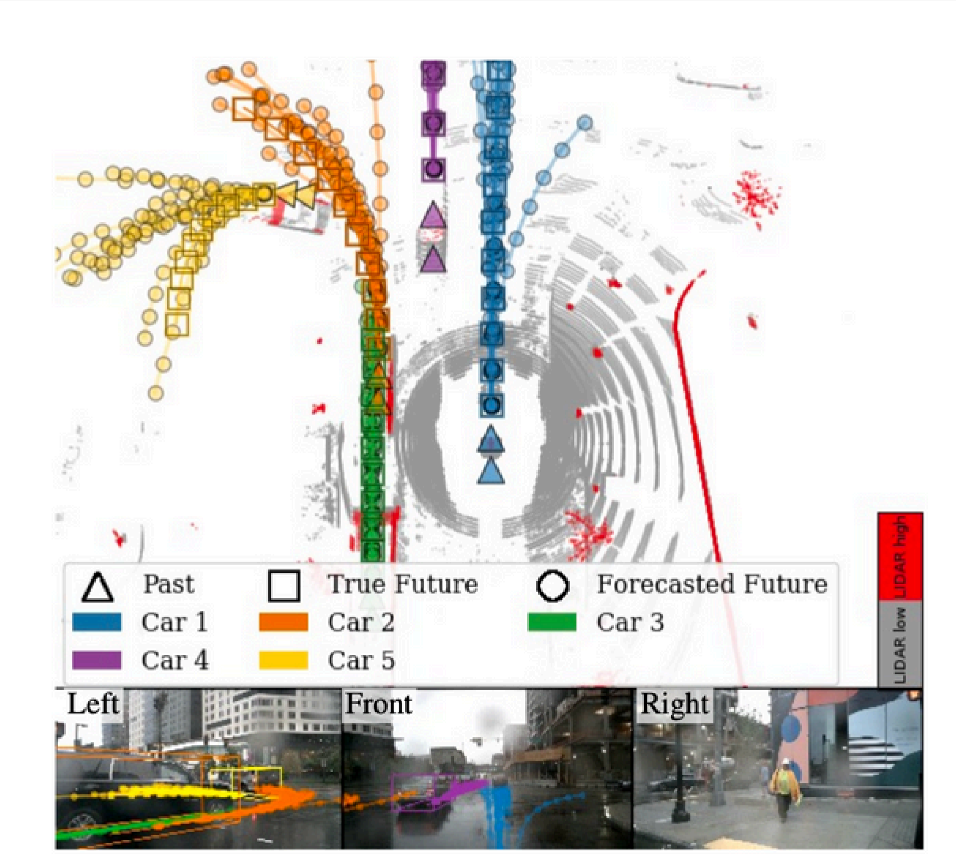
Félix DOUBLET

Papers survey

M1 research

"PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings" [Rinhart+ (Carnegie Mellon Univ) , ICCV19]

- **Objective**: Develop a probabilistic forecasting model to predict future interactions between multiple agents (vehicles) by understanding the goals of a controlled agent, in this case, an Autonomous Vehicle.
- **Methodology**: The model uses real and simulated data to forecast vehicle trajectories based on past positions and LIDAR data. It performs both standard forecasting and conditional forecasting, which predicts how all agents will likely respond to the goal of a controlled agent.
- **Model Implementation and Experiments**: The paper describes the implementation of a factorized flow-based generative model called "Estimating Social-forecast Probabilities" (ESP) and a goal-conditioned forecasting method called "Prediction Conditioned on Goals" (PRECOG).



Forecasting on nuScenes [4]. The input to our model is a high-dimensional LIDAR observation, which informs a distribution over all agents' future trajectories.

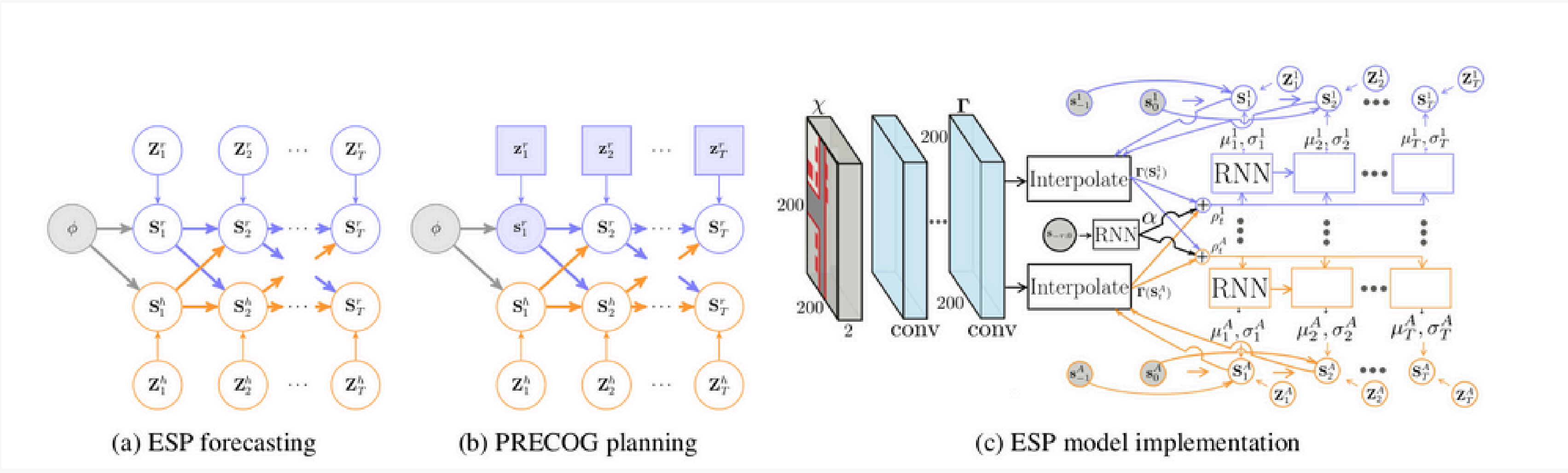
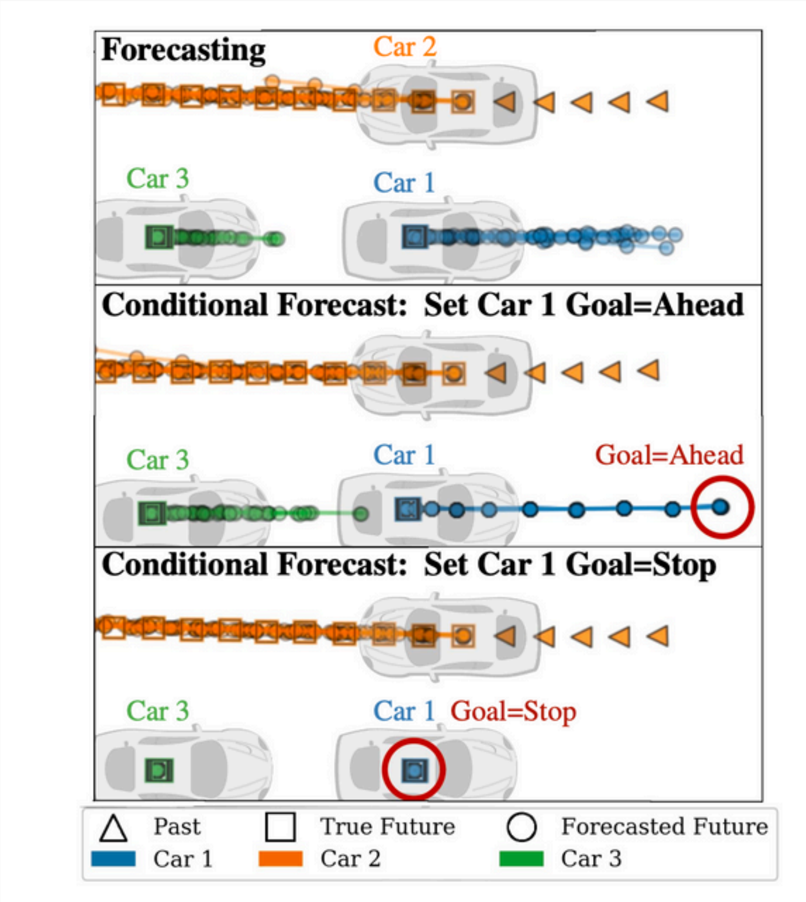


Figure 3: Our factorized latent variable model of forecasting and planning shown for 2 agents. In Fig. 3a our model uses latent variable $\mathbf{Z}^a_{t+1}$ to represent variation in agent a's *plausible* scene-conditioned reactions to all agents $\mathbf{S}_t$, causing uncertainty in every agents' future states $\mathbf{S}$. Variation exists because of unknown driver goals and different driving styles observed in the training data. Beyond forecasting, our model admits *planning* robot decisions by *deciding* $\mathbf{Z}^r = \mathbf{z}^r$ (Fig. 3b). Shaded nodes represent observed or determined variables, and square nodes represent robot decisions [2]. Thick arrows represent grouped dependencies of *non-Makovian* $\mathbf{S}_t$ "carried forward" (a regular edge exists between any pair of nodes linked by a chain of thick edges). Note $\mathbf{Z}$ *factorizes* across agents, isolating the robot's reaction variable $\mathbf{z}^r$. Human reactions remain uncertain ($\mathbf{Z}^h$ is unobserved) and uncontrollable (the robot cannot decide $\mathbf{Z}^h$), and yet the robot's decisions $\mathbf{z}^r$ will still *influence* human drivers $\mathbf{S}^h_{2:T}$ (and vice-versa). Fig. 3c shows our implementation. See Appendix C for details.



Conditioning the model on different Car 1 goals produces different predictions: here it forecasts Car 3 to move if Car 1 yields space, or stay stopped if Car 1 stays stopped.

RHINEHART, Nicholas, MCALLISTER, Rowan, KITANI, Kris, Sergey LEVINE. PRECOG: PREdictions Conditioned On Goals in Visual Multi-Agent Scenarios. ICCV (2019), **code**, **link**

"A Generalist Agent" (GATO) [Reed+ (DeepMind), Transactions on Machine Learning Research (11/2022)]

- **Approach**: Gato, the agent, can perform a wide range of tasks, including playing Atari games, captioning images, chatting, and manipulating objects with a robotic arm. It decides what outputs to generate (text, joint torques, button presses, etc.) based on its context.
- **Methodology**: The model uses a transformer neural network similar to large-scale language models. It's trained to handle various data types by serializing them into tokens, including images, text, proprioception, joint torques, button presses, and other actions.
- **Performance Analysis**: It demonstrates Gato's capabilities in various simulated control tasks, robotics, and text samples, showing that the model can achieve impressive results across different domains.

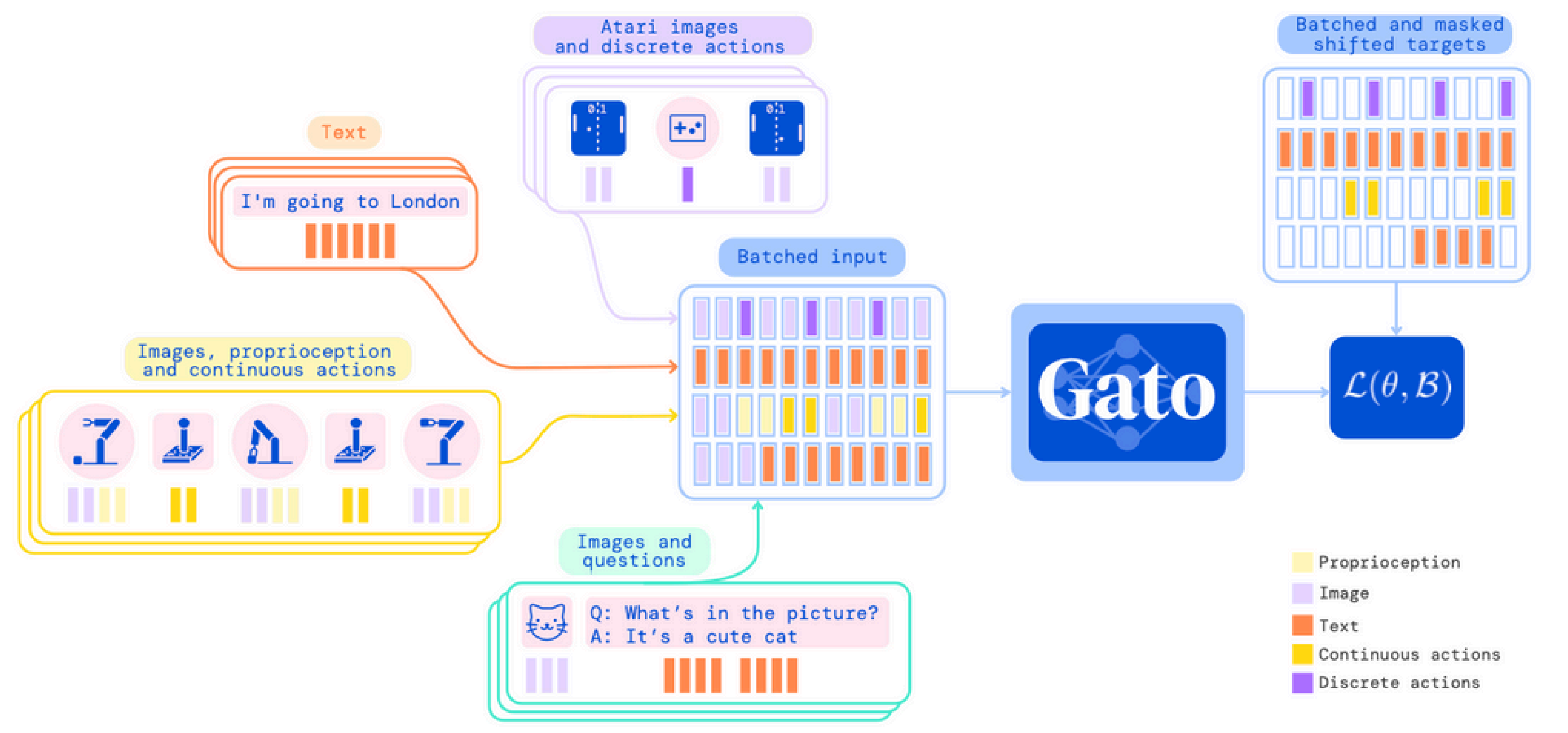


Figure 2: **Training phase of Gato**. Data from different tasks and modalities is serialized into a flat sequence of tokens, batched, and processed by a transformer neural network akin to a large language model. Masking is used such that the loss function is applied only to target outputs, i.e. text and various actions.

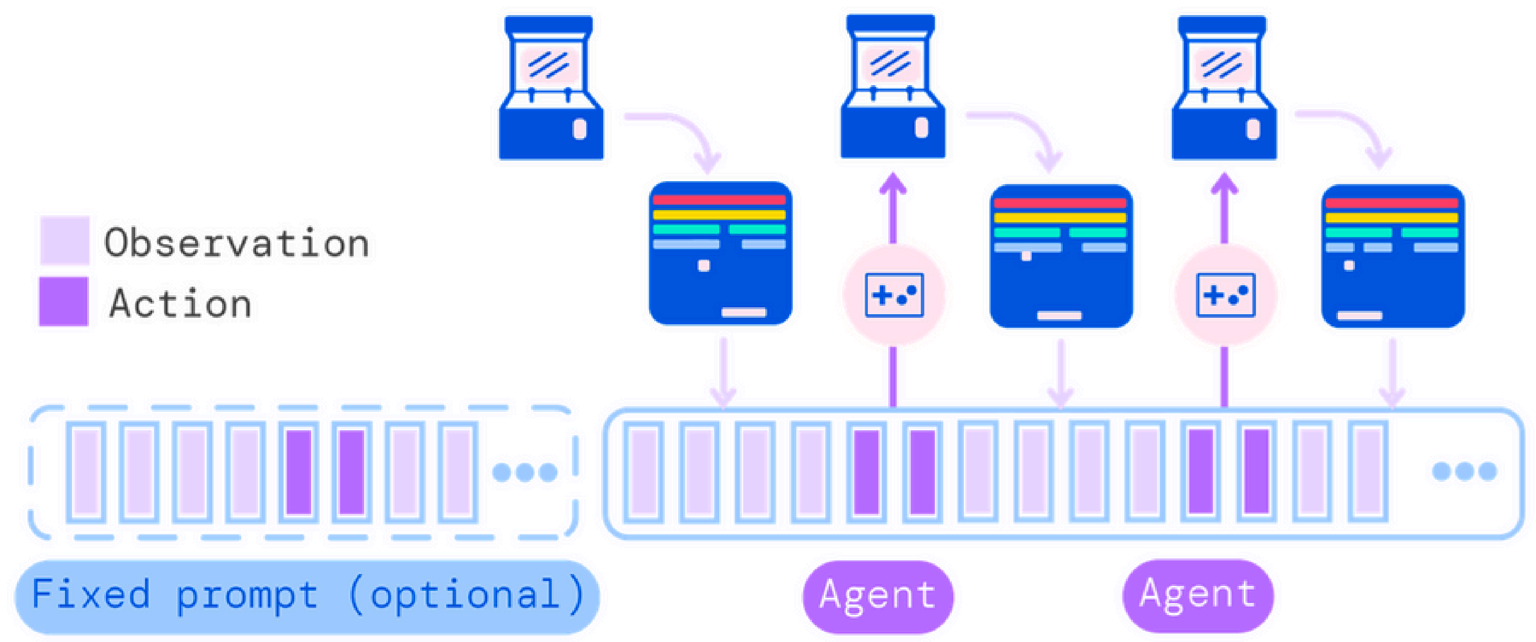


Figure 3: **Running Gato as a control policy.** Gato consumes a sequence of interleaved tokenized observations, separator tokens, and previously sampled actions to produce the next action in standard autoregressive manner. The new action is applied to the environment – a game console in this illustration, a new set of observations is obtained, and the process repeats.

REED, Scott, ZOLNA, Konrad, PARISOTTO, Emilio, et al. A generalist agent, arXiv preprint arXiv:2205.06175, 2022

unofficial code, link

"Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm" [Silver+ (DeepMind), arxiv preprint17]

- **Objective**: The development of AlphaZero, a single algorithm capable of learning and achieving superhuman performance in chess, shogi, and Go through self-play, without domain knowledge other than the game rules.
- **Methodology**: AlphaZero uses a deep neural network and a general-purpose Monte Carlo tree search (MCTS) algorithm. It learns strategies and tactics entirely from self-play without prior knowledge.
- **Results**: Within 24 hours, AlphaZero achieved a superhuman level of play in all three games and convincingly defeated world-champion programs in each case.
- **General Reinforcement Learning Algorithm:** AlphaZero represents a significant shift from specialized, domain-specific AI to a more generalist approach. It does not rely on historical data but learns from scratch through self-play.
- **Deep Neural Networks and MCTS**: AlphaZero combines deep neural networks with a powerful search algorithm (MCTS), enabling it to evaluate and decide on the most promising moves in complex game scenarios.

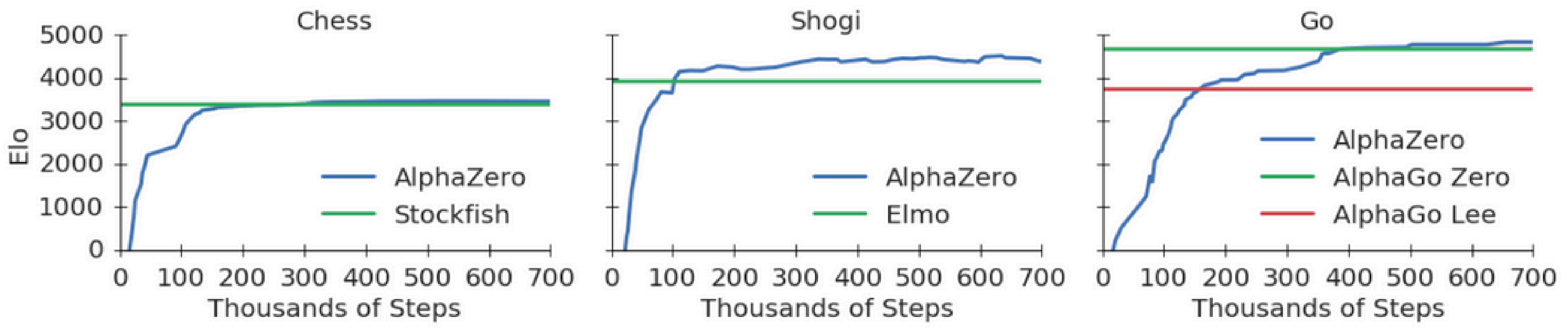


Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (*29*).

SILVER, David, HUBERT, Thomas, SCHRITTWIESER, Julian, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm,arXiv preprint arXiv:1712.01815, 2017, **unofficial code**, **link**

"Building High-level Features Using Large Scale Unsupervised Learning," [Quoc V. LE+ (Google), ICML13]

- **Objective**: The paper explores building high-level, class-specific feature detectors from only unlabeled data. It aims to answer whether it is possible to learn a face detector using only unlabeled images.
- **Methodology**: The authors trained a 9-layered locally connected sparse auto encoder with pooling and local contrast normalization on a large dataset of 10 million 200x200 pixel images downloaded from the Internet.
- **Results**: The results showed that it is possible to train a feature detector for faces without labeled images. The model was robust to translation, scaling, and out-of-plane rotation and was also sensitive to other high-level concepts like cat faces and human bodies.

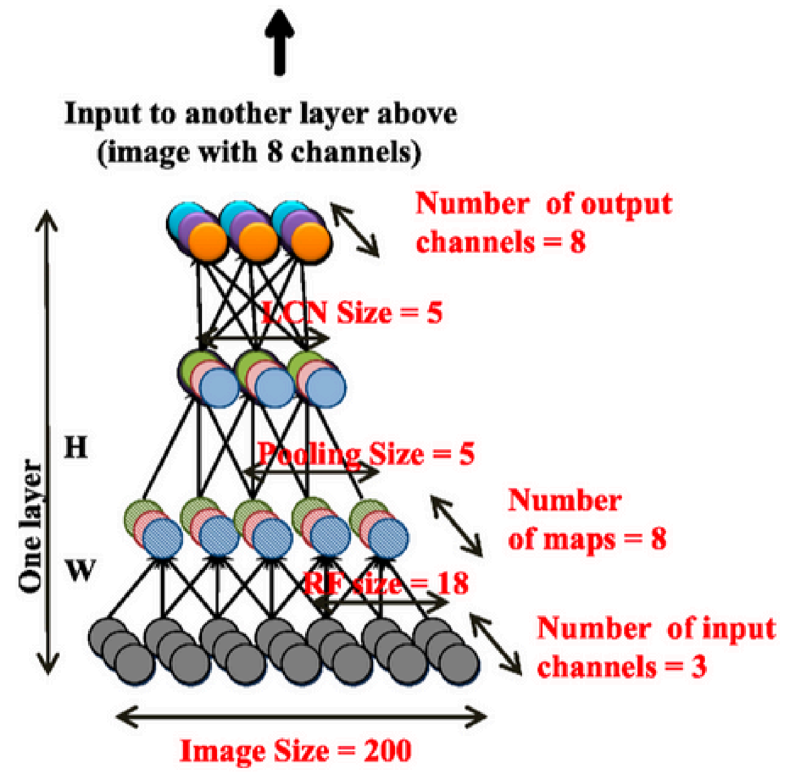


Figure 1. The architecture and parameters in one layer of our network. The overall network replicates this structure three times. For simplicity, the images are in 1D.

Figure 3. Top: Top 48 stimuli of the best neuron from the test set. Bottom: The optimal stimulus according to numerical constraint optimization.

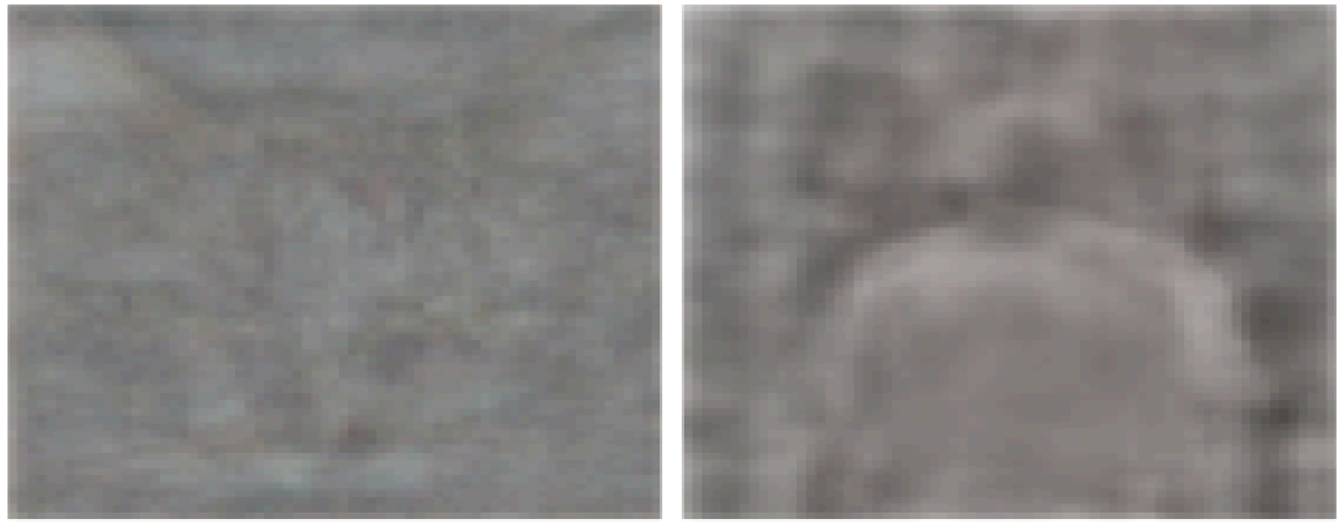


Figure 6. Visualization of the cat face neuron (left) and human body neuron (right).

LE, Quoc V. Building high-level features using large scale unsupervised learning ICML, 2013. p. 8595-8598

link

Solving Challenging Math Word Problems Using GPT-4 Code Interpreter with Code-based Self-verification [Zhou+ (MMLab), arxiv preprint2023]

- **Objective**: The study aims to improve the reasoning capability of GPT-4 in solving challenging math problems by introducing constraints on Code Usage Frequency and implementing a code-based self-verification (CSV) method.
- **Methodology**: The paper explores GPT-4's ability to generate and execute code, evaluate the output of code execution, and rectify its solution. The authors propose the CSV method, which uses zero-shot prompts to encourage GPT-4 to self-verify its answers using code.
- **Results**: This approach significantly boosts the model's accuracy in solving math problems, as demonstrated by a dramatic increase in zero-shot accuracy on the MATH dataset (from 53.9% to 84.3%).

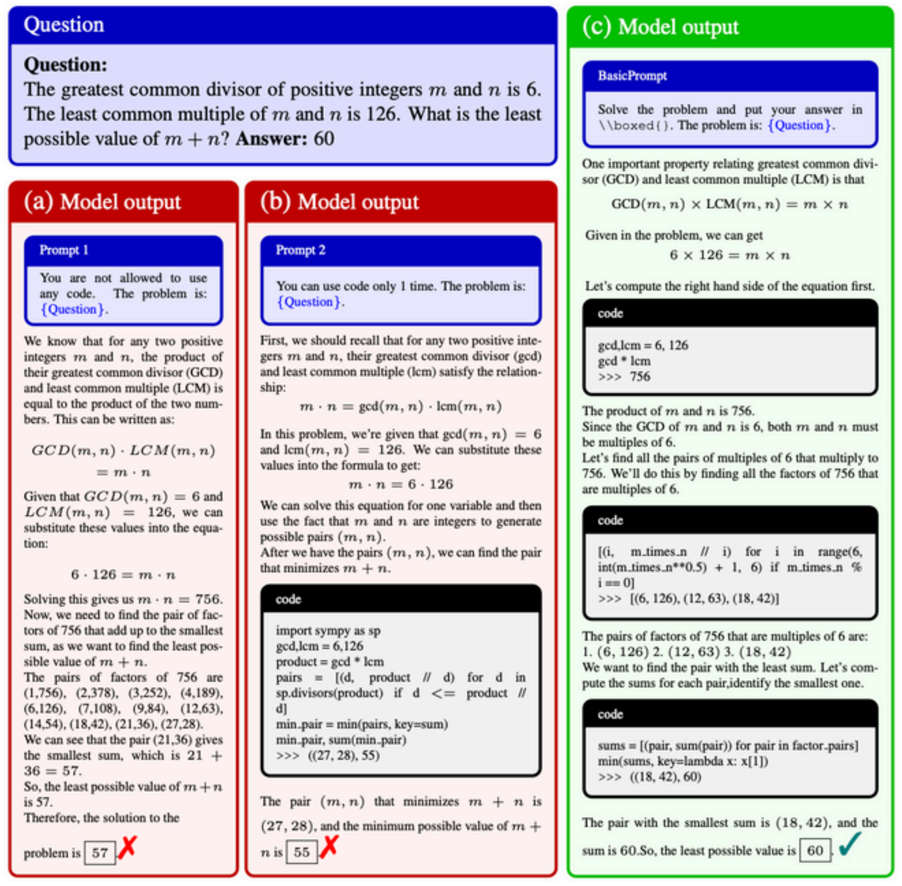


Figure 1: Example from the 277th number theory problem in the MATH dataset: (a) Prompt 1 generates intermediate text-only reasoning steps. (b) Prompt 2 generates intermediate text reasoning steps and post-hoc Python code, akin to PAL (Gao et al., 2023). (c) Basic prompt that set no constrain on code usage. Details of the three prompts are presented in Sec. 3.1. (For more examples, see Tab. 9 and Tab. 10.)

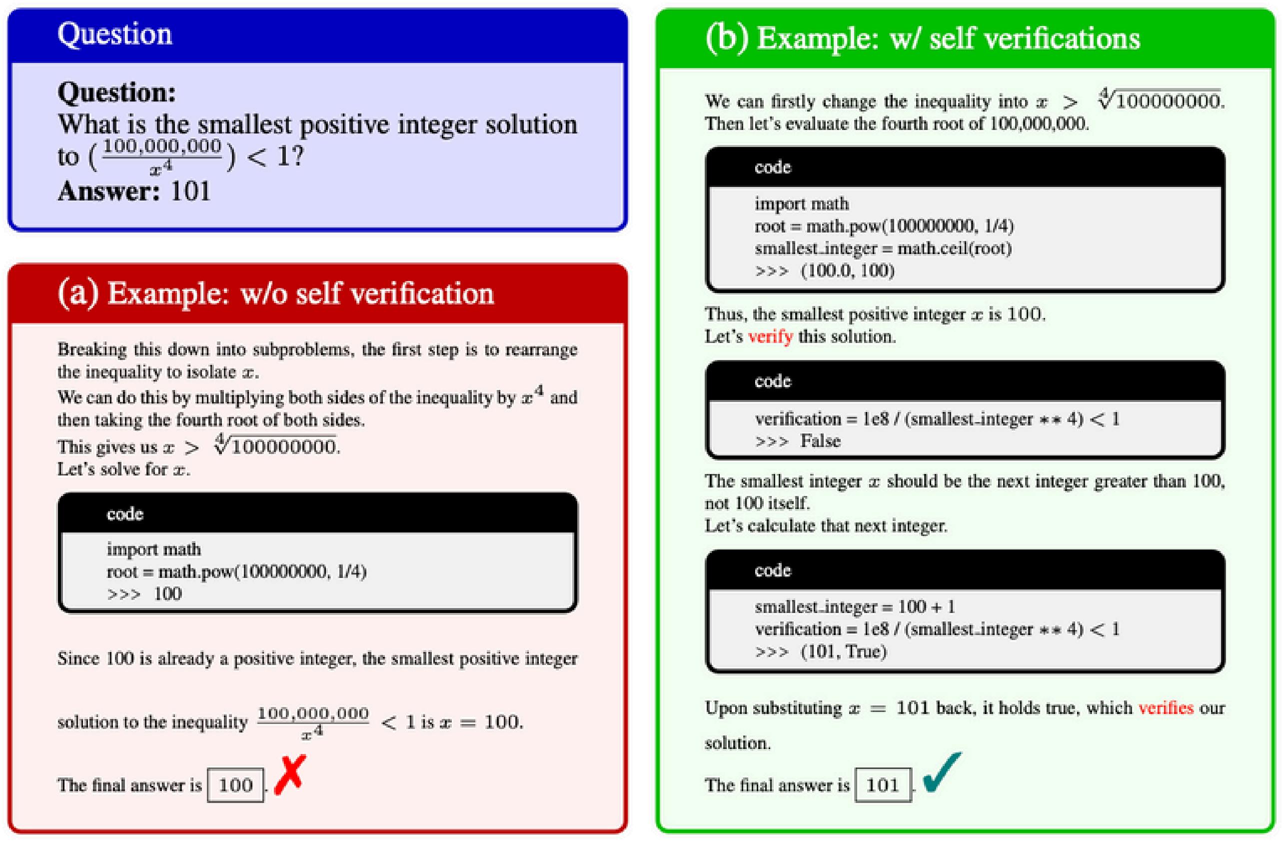


Figure 3: Question from the 712th intermediate algebra problem in the MATH dataset. (a) Without self-verification, the model generates a wrong answer. (b) With self-verification, the model corrects the error and generates the correct answer. The CSV prompt: *To solve the problem using code interpreter step by step, and please verify your answer using code interpreter.*

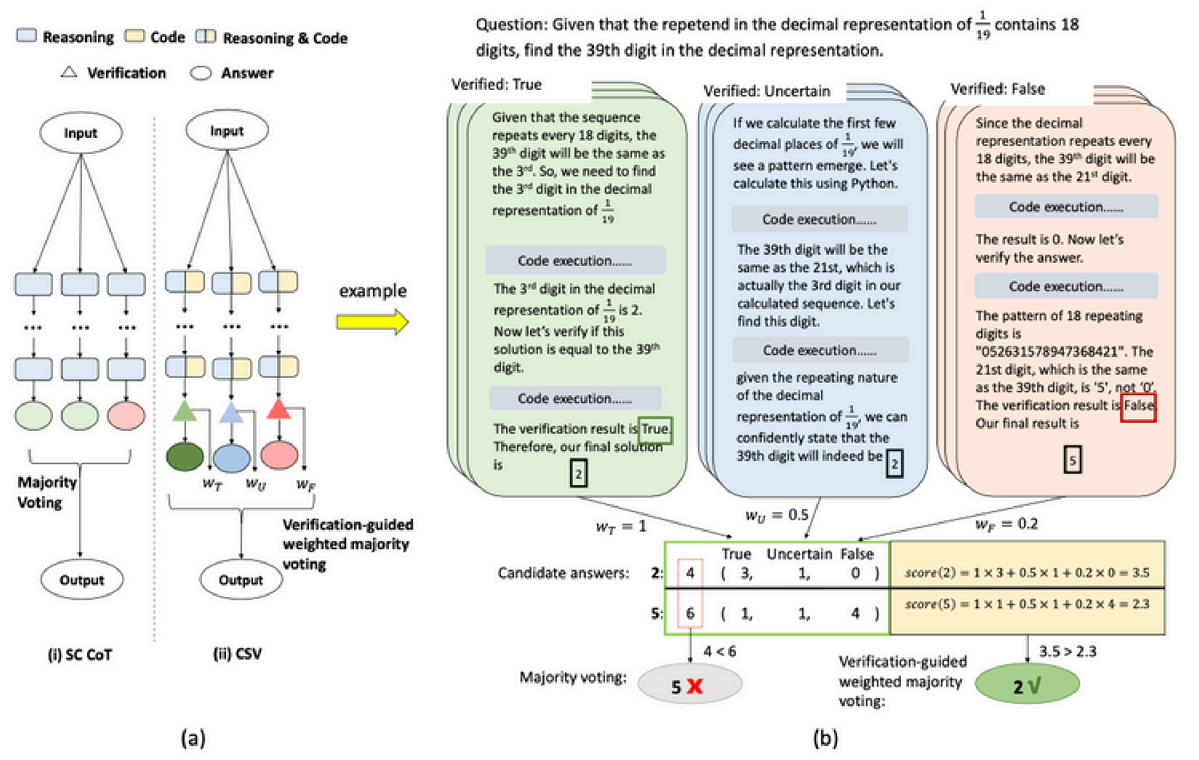


Figure 4: (a) Illustration of the Naive majority voting (Wang et al., 2023) and our Verification-guided weighted majority voting. The full pipeline of the proposed Verification-guided Weighted Majority Voting framework. We use the model to generate several different solutions. Then we detect the self-verification state of each solution, and classify them into three states: *True*, *Uncertain*, and *False*. According to the state of the verification, we assign each solution a different weight, and use the classified result to vote the score of each possible answer.

ZHOU, Aojun, WANG, Ke, LU, Zimu, et al. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. arXiv preprint arXiv:2308.07921, 2023, link

GPT4 Technical Report [OpenAI, 2023]

- **Description**: GPT-4 is equipped with capabilities to understand and generate responses related to mathematical problems. It can interpret and solve a wide range of mathematical questions, from basic arithmetic to more complex fields such as calculus, algebra, and statistics. The model is designed to process textual descriptions of mathematical problems and provide solutions or explanations in a human-readable format.
- **Accuracy Concerns**: While GPT-4 is generally reliable, it is not infallible and can sometimes provide incorrect or misleading solutions, necessitating cross-verification.
- **Lack of Intuition**: The model does not possess human-like intuition or understanding, which can be crucial in solving more nuanced or abstract mathematical problems.

Exam	GPT-4	GPT-4 (no vision)	GPT-3.5
Uniform Bar Exam (MBE+MEE+MPT)	298 / 400 (~90th)	298 / 400 (~90th)	213 / 400 (~10th)
LSAT	163 (~88th)	161 (~83rd)	149 (~40th)
SAT Evidence-Based Reading & Writing	710 / 800 (~93rd)	710 / 800 (~93rd)	670 / 800 (~87th)
SAT Math	700 / 800 (~89th)	690 / 800 (~89th)	590 / 800 (~70th)
Graduate Record Examination (GRE) Quantitative	163 / 170 (~80th)	157 / 170 (~62nd)	147 / 170 (~25th)
Graduate Record Examination (GRE) Verbal	169 / 170 (~99th)	165 / 170 (~96th)	154 / 170 (~63rd)
Graduate Record Examination (GRE) Writing	4 / 6 (~54th)	4 / 6 (~54th)	4 / 6 (~54th)
USABO Semifinal Exam 2020	87 / 150 (99th - 100th)	87 / 150 (99th - 100th)	43 / 150 (31st - 33rd)
USNCO Local Section Exam 2022	36 / 60	38 / 60	24 / 60
Medical Knowledge Self-Assessment Program	75 %	75 %	53 %
Codeforces Rating	392 (below 5th)	392 (below 5th)	260 (below 5th)
AP Art History	5 (86th - 100th)	5 (86th - 100th)	5 (86th - 100th)
AP Biology	5 (85th - 100th)	5 (85th - 100th)	4 (62nd - 85th)
AP Calculus BC	4 (43rd - 59th)	4 (43rd - 59th)	1 (0th - 7th)
AP Chemistry	4 (71st - 88th)	4 (71st - 88th)	2 (22nd - 46th)
AP English Language and Composition	2 (14th - 44th)	2 (14th - 44th)	2 (14th - 44th)
AP English Literature and Composition	2 (8th - 22nd)	2 (8th - 22nd)	2 (8th - 22nd)
AP Environmental Science	5 (91st - 100th)	5 (91st - 100th)	5 (91st - 100th)
AP Macroeconomics	5 (84th - 100th)	5 (84th - 100th)	2 (33rd - 48th)
AP Microeconomics	5 (82nd - 100th)	4 (60th - 82nd)	4 (60th - 82nd)
AP Physics 2	4 (66th - 84th)	4 (66th - 84th)	3 (30th - 66th)
AP Psychology	5 (83rd - 100th)	5 (83rd - 100th)	5 (83rd - 100th)
AP Statistics	5 (85th - 100th)	5 (85th - 100th)	3 (40th - 63rd)
AP US Government	5 (88th - 100th)	5 (88th - 100th)	4 (77th - 88th)
AP US History	5 (89th - 100th)	4 (74th - 89th)	4 (74th - 89th)
AP World History	4 (65th - 87th)	4 (65th - 87th)	4 (65th - 87th)
AMC 10[3]	30 / 150 (6th - 12th)	36 / 150 (10th - 19th)	36 / 150 (10th - 19th)
AMC 12[3]	60 / 150 (45th - 66th)	48 / 150 (19th - 40th)	30 / 150 (4th - 8th)
Introductory Sommelier (theory knowledge)	92 %	92 %	80 %
Certified Sommelier (theory knowledge)	86 %	86 %	58 %
Advanced Sommelier (theory knowledge)	77 %	77 %	46 %
Leetcode (easy)	31 / 41	31 / 41	12 / 41
Leetcode (medium)	21 / 80	21 / 80	8 / 80
Leetcode (hard)	3 / 45	3 / 45	0 / 45

Table 1. GPT performance on academic and professional exams. In each case, we simulate the conditions and scoring of the real exam. We report GPT-4's final score graded according to exam-specific rubrics, as well as the percentile of test-takers achieving GPT-4's score.

OpenAI, GPT-4 Technical Report. 2023, link

MINEDOJO: Building Open-Ended Embodied Agents with Internet-Scale Knowledge [Fan+ (NVIDIA), NeurIPS22]

- **Objective**: VOYAGER is developed to continuously explore the Minecraft world, acquire diverse skills, and make novel discoveries without human intervention.
- **Methodology**: An automatic curriculum maximizing exploration. An ever-growing skill library of executable code for storing and retrieving complex behaviors. A novel iterative prompting mechanism that incorporates environment feedback, execution errors, and self-verification for program improvement.
- **Interaction with GPT-4**: VOYAGER interacts with GPT-4 via black box queries, eliminating the need for model parameter fine-tuning.
- **Results:** It achieves significant milestones, such as obtaining 3.3 times more unique items, traveling 2.3 times longer distances, and unlocking key tech tree milestones up to 15.3 times faster than previous state-of-the-art methods.

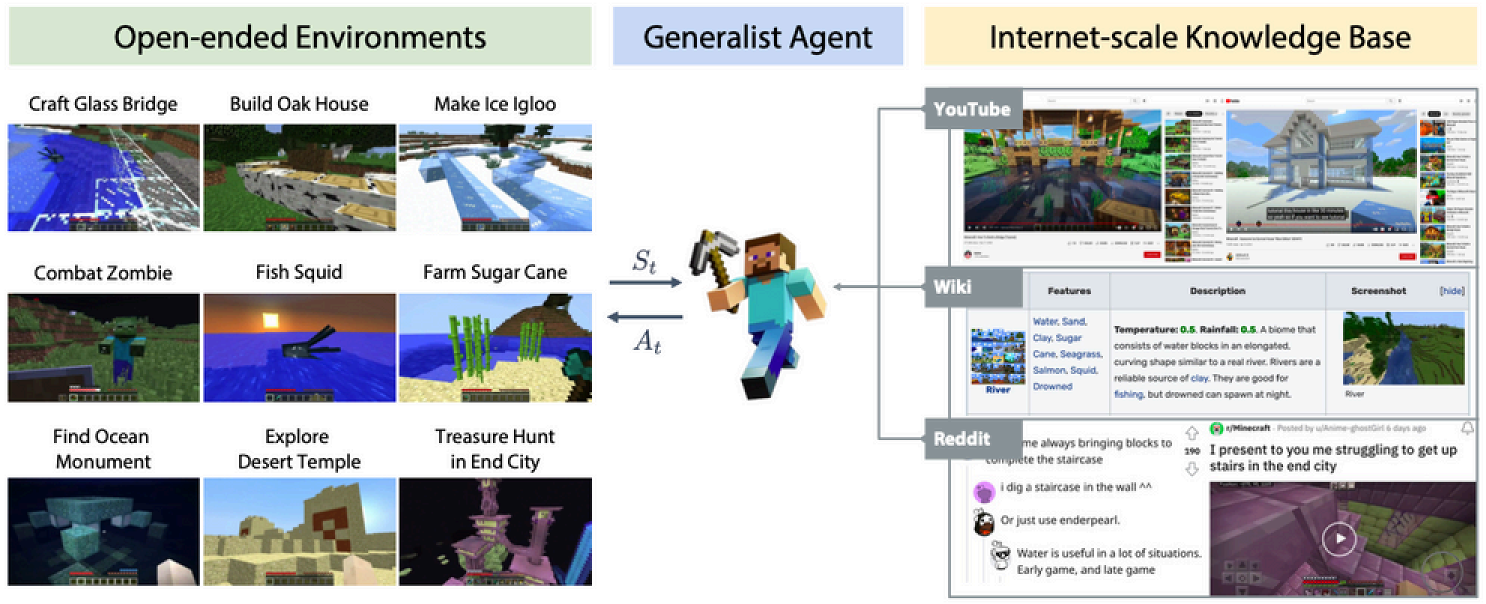


Figure 1: MINEDOJO is a novel framework for developing open-ended, generally capable agents that can learn and adapt continually to new goals. MINEDOJO features a benchmarking suite with ***thousands* of diverse open-ended tasks** specified in natural language prompts, and also provides an **internet-scale, multimodal knowledge base** of YouTube videos, Wiki pages, and Reddit posts. The database captures the collective experience and wisdom of millions of Minecraft gamers for an AI agent to learn from. Best viewed zoomed in.

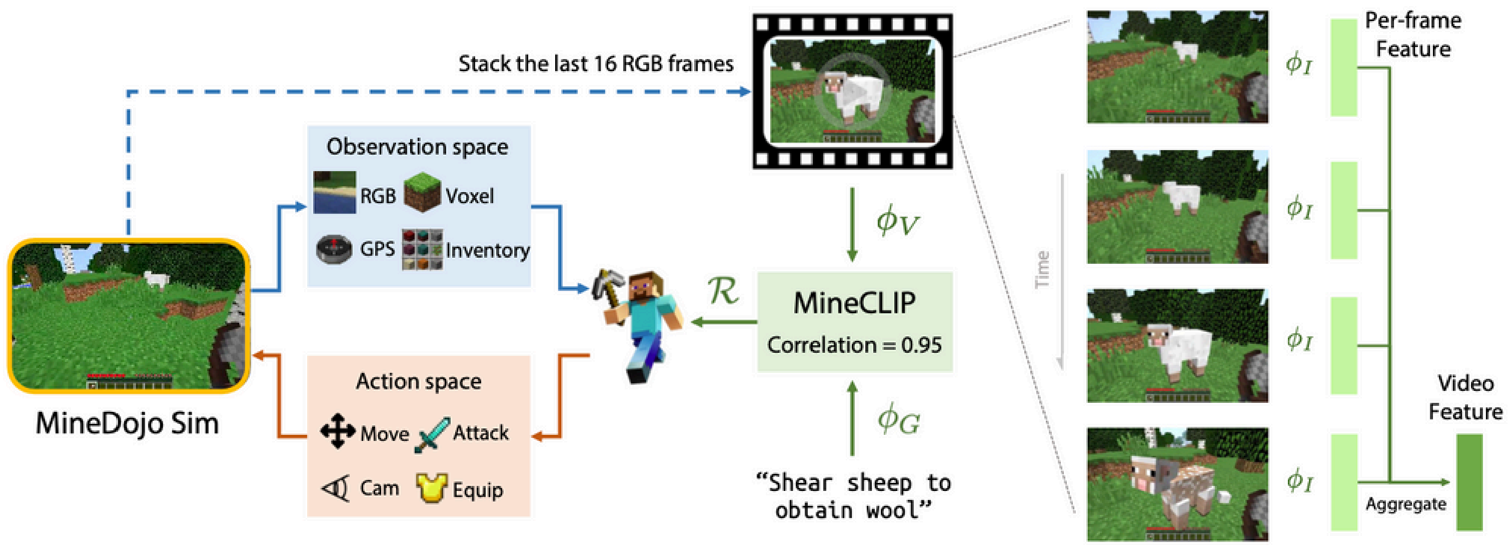


Figure 4: Algorithm design. MINECLIP is a contrastive video-language model pre-trained on MINEDOJO's massive Youtube database. It computes the correlation between an open-vocabulary language goal string and a 16-frame video snippet. The correlation score can be used as a learned dense reward function to train a strong multi-task RL agent.

FAN, Linxi, WANG, Guanzhi, JIANG, Yunfan, et al. Minedojo: Building open-ended embodied agents with internet-scale knowledge. Advances in Neural Information Processing Systems, 2022, vol. 35, p. 18343-18362., code, link

EUREKA: Human-Level Reward Design via Coding Large Language Models" [MA+ (NVIDIA), arXiv preprint23]

- **Objective**: The paper introduces EUREKA, an algorithm that uses coding LLMs to autonomously generate reward functions for RL tasks, aiming to achieve human-level performance in reward design without task-specific prompts or predefined templates.
- **Approach**: EUREKA leverages the code-writing and zero-shot generation capabilities of LLMs to perform evolutionary optimization over reward code, resulting in rewards that can be used to acquire complex skills through reinforcement learning.
- **Results**: The algorithm outperforms human experts in 83% of tasks across 29 open-source RL environments, showing an average normalized improvement of 52%. EUREKA also introduces a new gradient-free approach to reinforcement learning from human feedback (RLHF), improving the quality and safety of generated rewards.

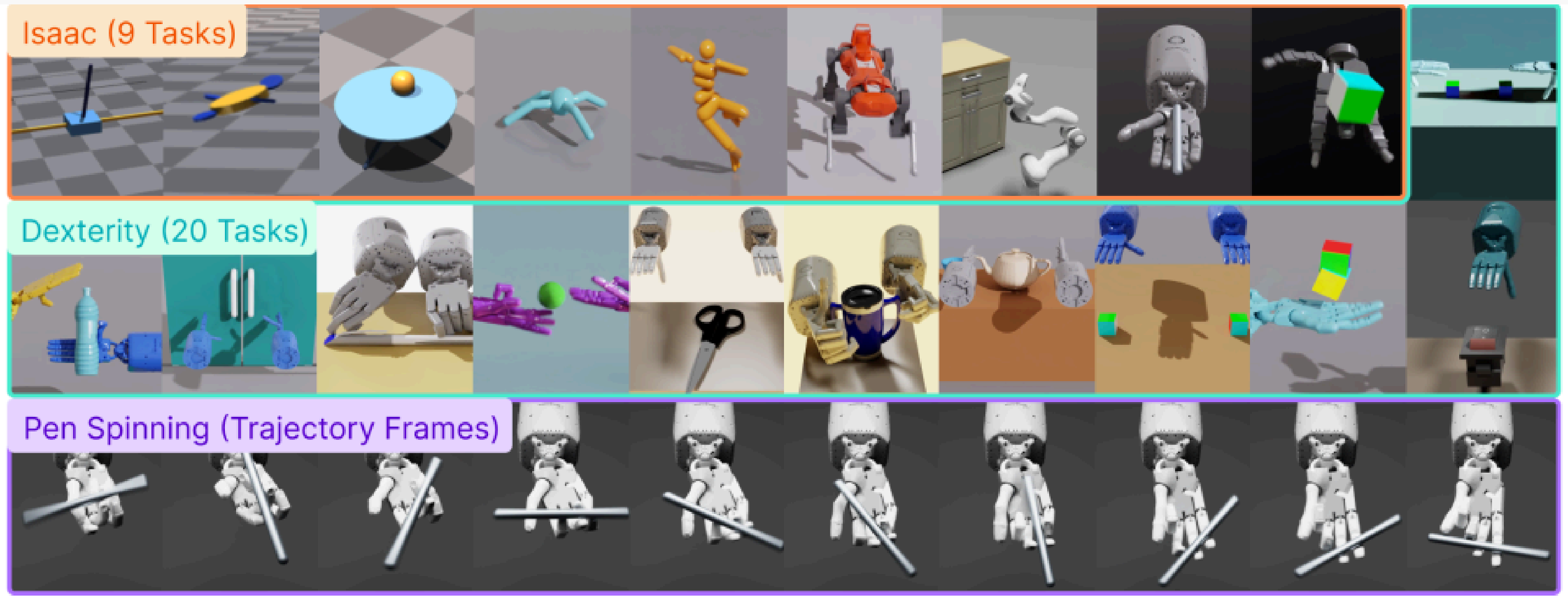


Figure 1: EUREKA generates human-level reward functions across diverse robots and tasks. Combined with curriculum learning, EUREKA for the first time, unlocks rapid pen-spinning capabilities on an anthropomorphic five-finger hand. Figures rendered using Omniverse (NVIDIA, 2023).

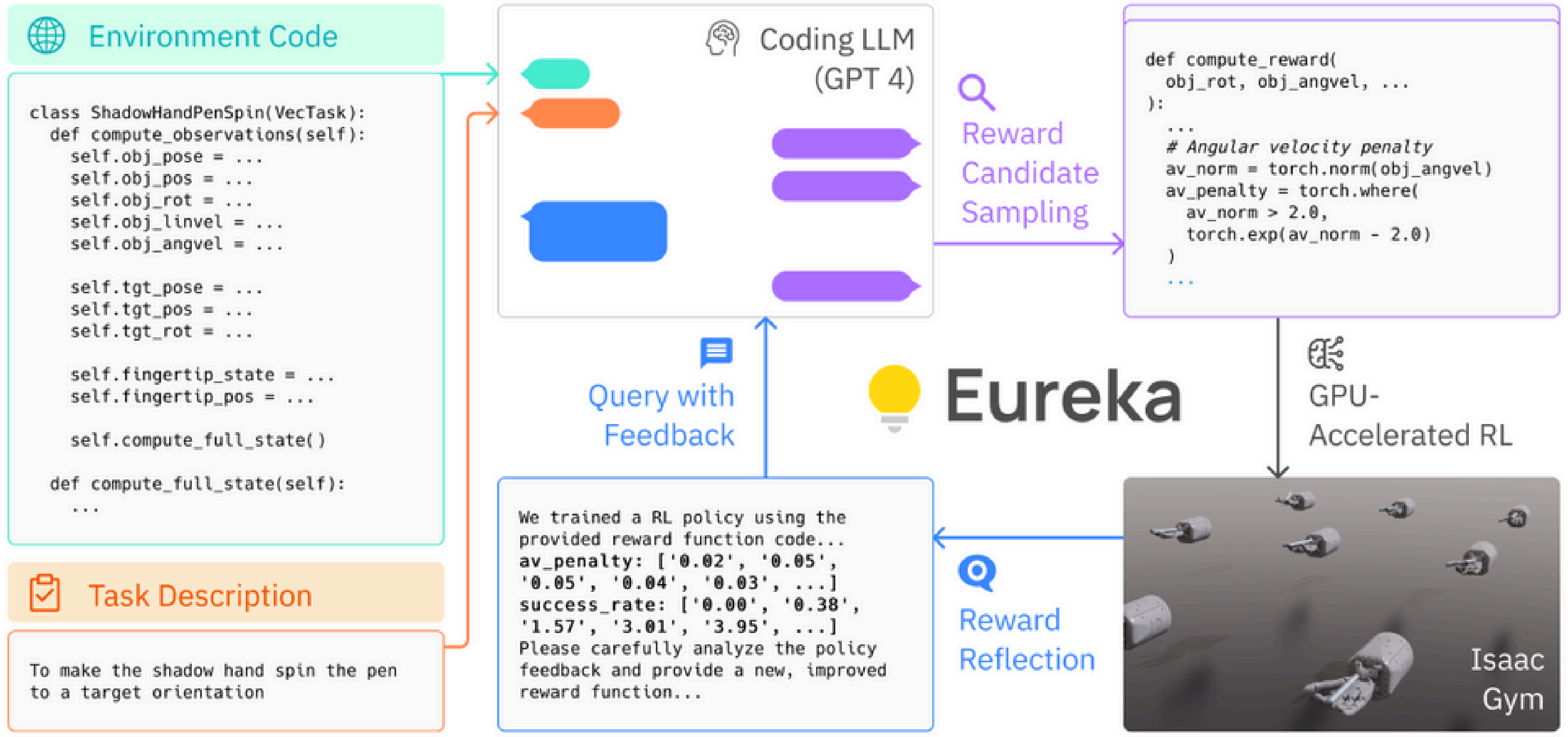


Figure 2: EUREKA takes unmodified environment source code and language task description as context to zero-shot generate executable reward functions from a coding LLM. Then, it iterates between reward sampling, GPU-accelerated reward evaluation, and reward reflection to progressively improve its reward outputs.

MA, Yecheng Jason, LIANG, William, WANG, Guanzhi, et al.Eureka: Human-level reward design via coding large language models. arXiv preprint arXiv:2310.12931, 2023., code, link

Learning to Control Self-Assembling Morphologies: A Study of Generalization via Modularity [Pathak+ (Berkeley), NeurIPS19]

- **Objective**: Primitive agents, or limbs, can link up to solve tasks. Each limb has a neural net controlling the torque applied to its joints, and they can dynamically link and unlink, changing the agent's shape.
- **Approach**: The policy architecture of these self-assembling agents is represented via dynamic graph networks, aligning with the agent's morphology.
- **Modular Co-Evolution**: Investigating the co-evolution of control and morphology, the research uses a minimalist design for agents and environments, ensuring natural emergence of complex morphologies.

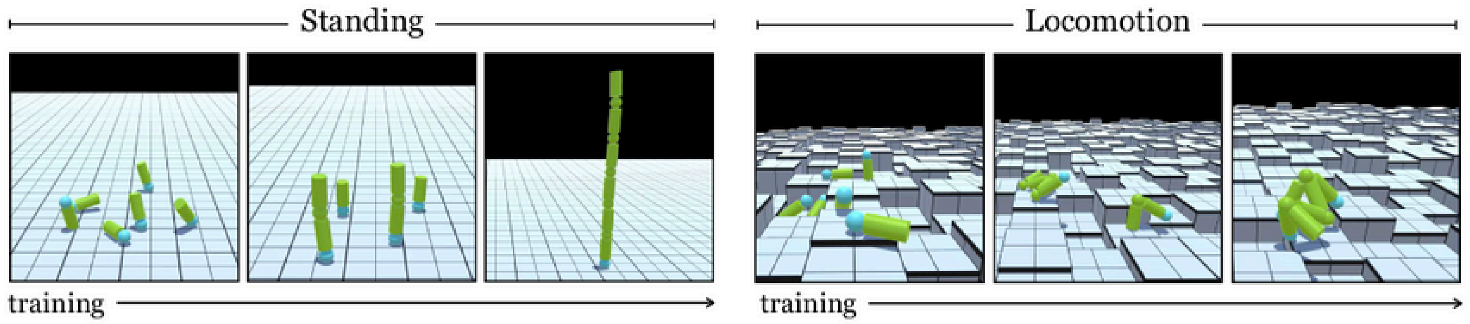


Figure 3: Co-evolution of Morphology w/ Control during Training: The gradual co-evolution of controller as well as the morphology of self-assembling agents over the course of training for the task of Standing Up (left) and Locomotion (right).

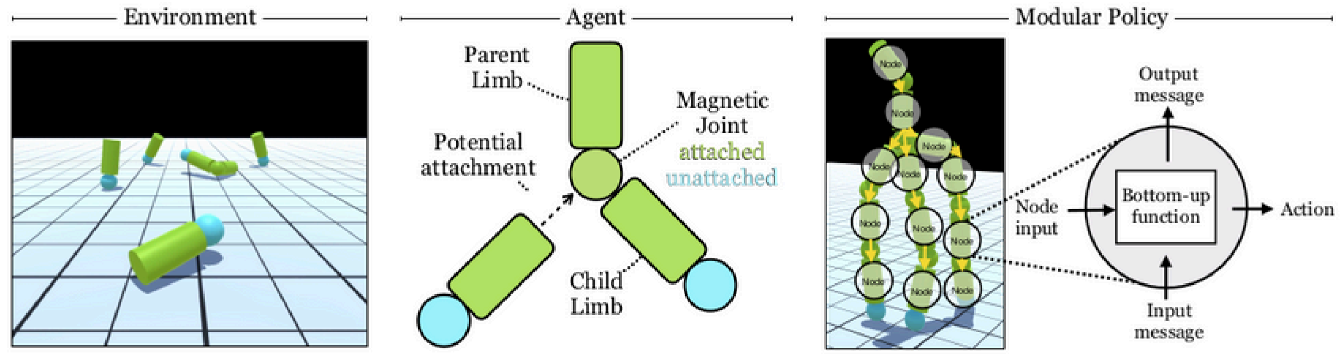


Figure 1: This work investigates the joint learning of control and morphology in self-assembling agents. Several primitive agents, containing a cylindrical body with a configurable motor, are dropped in a simulated environment (left). These primitive agents can self-assemble into collectives using magnetic joints (middle). The policy of the self-assembled agent is represented via proposed dynamic graph networks (DGN) with shared parameters (modular) across each limb (right).

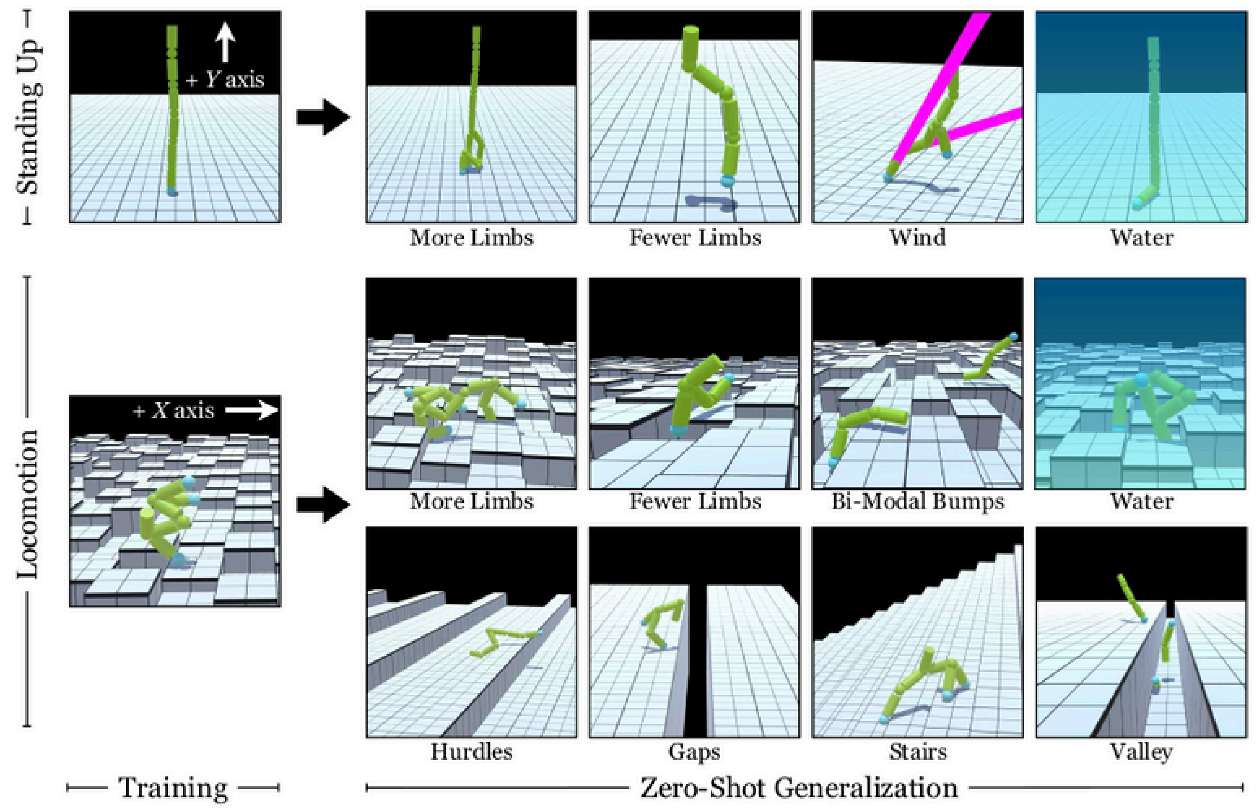


Figure 2: We illustrate our dynamic agents in two environments / tasks: standing up and locomotion. For each of these, we generate several new environment for evaluating generalization. Refer to project video at https://pathak22.github.io/modular-assemblies/ for better understanding of tasks.

PATHAK, Deepak, LU, Christopher, DARRELL, Trevor, et al.Learning to control self-assembling morphologies: a study of generalization via modularity. Advances in Neural Information Processing Systems, 2019, vol. 32., , [code](), [link]()

The Sensory Neuron as a Transformer: Permutation-Invariant Neural Networks for Reinforcement Learning [Tang+ (GoogleBrain), NeurIPS21]

- **Objective:** The paper explores artificial systems that can adapt to sensory substitutions without retraining. It focuses on the development of sensory networks that can integrate information received locally and collectively produce a globally coherent policy. These systems are also capable of handling random permutations of their inputs during an episode.
- **Methodology :** The concept of sensory substitution is explored, where the brain uses one sensory modality to supply environmental information typically gathered by another sense. The paper aims to create artificial systems that rapidly adapt to such sensory substitutions.
- **Sensory Neurons with Attention** : The approach involves feeding each sensory input into distinct neural networks without a fixed relationship with one another. An attention mechanism is employed to allow these networks to communicate and form a coherent global policy.
- **Results:** Experiments are conducted on different Reinforcement Learning (RL) environments to study the properties of permutation-invariant RL agents. These include tasks like Cart-pole swing up, PyBullet Ant, Atari Pong, and CarRacing.

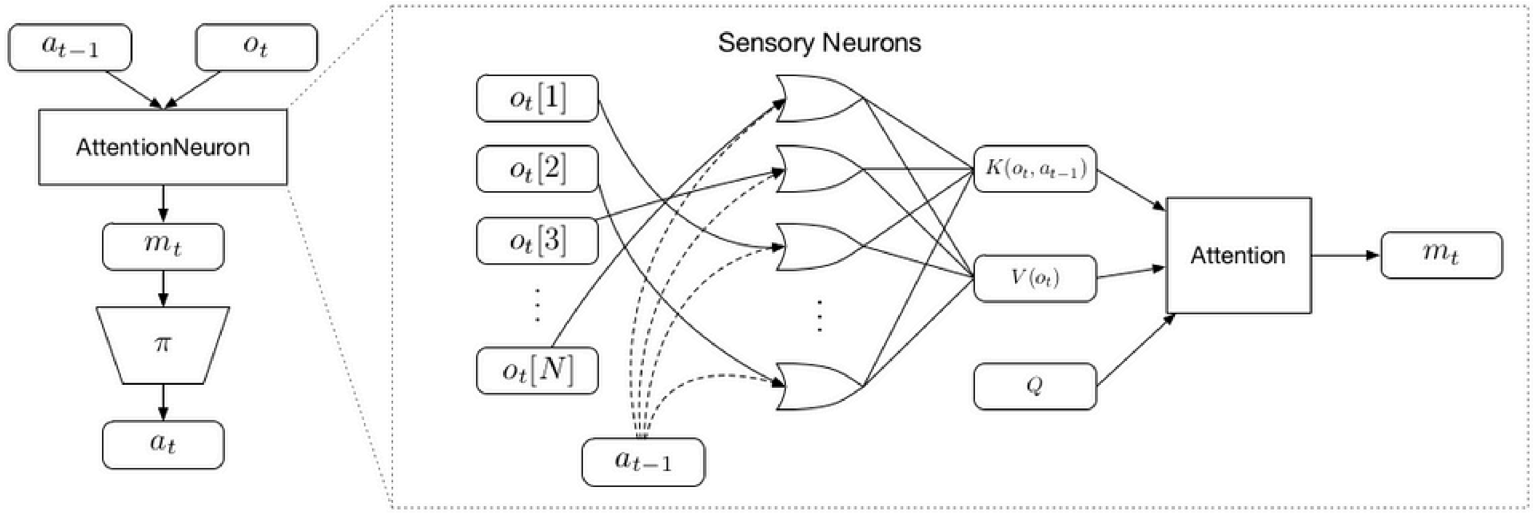


Figure 2: *Overview of Method.* AttentionNeuron is a standalone layer, in which each sensory neuron only has access to a part of the unordered observations o_t. Together with the agent's previous action a_{t-1}, each neuron generates messages independently using the shared functions $f_k(o_t[i], a_{t-1})$ and $f_v(o_t[i])$. The attention mechanism summarizes the messages into a global latent code m_t.

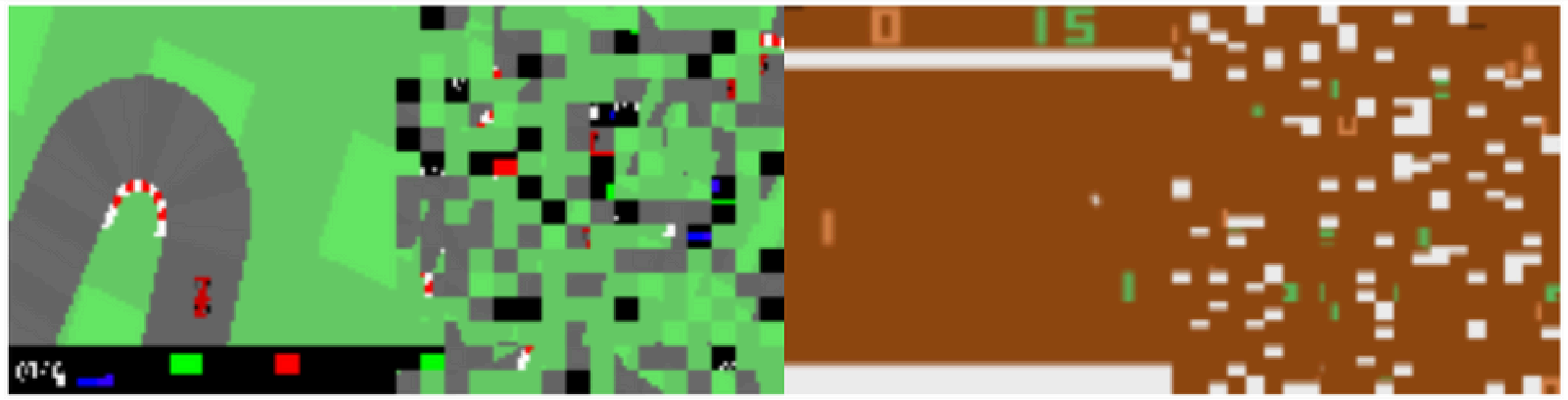


Figure 1: *Comparison of visual input intended for the game player, and what our system receives.* We partition the visual input from CarRacing (Left) and Atari Pong (right) into a 2D grid of small patches, and randomly permute their ordering. Each sensory neuron in the system receives a stream of visual input at a particular permuted patch location, and through coordination, must complete the task at hand, even if the visual ordering is randomly permuted again several times during an episode.

"Meta-Learning Bidirectional Update Rules [Sandler+ (Google), ICML21]

- **Objective:** This paper introduces a generalized neural network with multiple neuron and synapse states. It shows that traditional gradient-based backpropagation can be seen as a two-state network case. The paper proposes a framework where networks don't rely on gradients but use a bidirectional Hebb-style update rule, parameterized by a shared low-dimensional "genome."
- **Methodology :** The authors propose learning the rules for both forward and back-propagation of neuron activation from scratch, introducing a generalization where neurons can have multiple states.
- **BLUR (Bidirectional Learned Update Rules) :** Describes a set of multi-state update rules, enabling networks to learn new tasks without explicit gradients. The meta-learned "genomes" can train networks on unseen tasks faster than gradient-based methods.
- **Meta-Learning the Genome**: The process involves meta-learning genomes that can solve classification problems with multiple hidden layers. The approach includes using both traditional optimization techniques and evolutionary strategies like CMA-ES (Covariance Matrix Adaptation Evolution Strategy).

Meta-Learning Bidirectional Update Rules

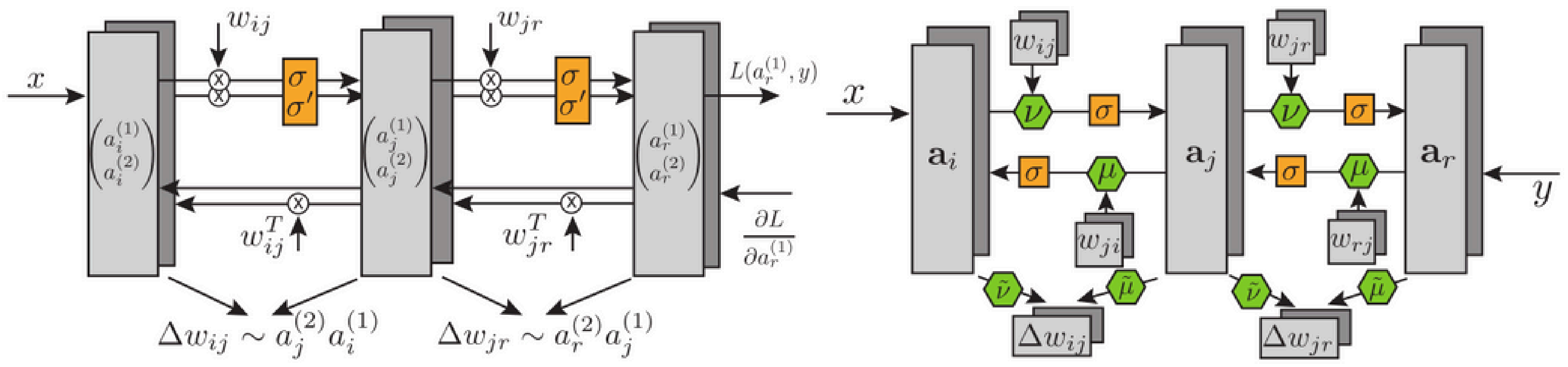


Figure 1. Generalization of a three layers feed-forward neural networks as a multi-state systems. *Left:* Forward pass and chain-rule backpropagation organized as a generalized two-state network. Arrows indicate the flow of information from forward and backward passes to synapse updates. *Right:* Our proposed generalized formulation. Green nodes, defined in the genome, control the amount of mixing between the states. They are fixed during the synapse update (inner-loop) and are optimized during the meta-training (outer-loop). Grey boxes indicate multi-state variables. Orange boxes represent activation functions. Notice the symmetry between forward and backward passes.

SANDLER, Mark, VLADYMYROV, Max, ZHMOGINOV, Andrey, et al. Meta-learning bidirectional update rules. In : International Conference on Machine Learning. PMLR, 2021. p. 9288-9300, **link**

A Zero-Shot Language Agent for Computer Control with Structured Reflection [Li+ (Google), EMNLP23]

- **Objective:** The paper introduces a zero-shot language agent capable of performing tasks in a live computer environment without needing expert traces or extensive training on specific tasks. This agent can plan executable actions in a partially observed environment and iteratively learn from its mistakes through self-reflection and structured thought management.
- **Methodology:** The paper builds upon previous works using large language models (LLMs) for action generation and task completion in various environments. It addresses the limitations of requiring expert traces for learning, proposing a zero-shot agent that adapts and learns autonomously.
- **Zero-Shot Learning Approach**: The agent uses PaLM2, a recent LLM, and a unified instruction prompt set across different tasks, avoiding the need for task-specific customization.
- **Structured Reflection and Thought Management:** The agent employs a reflection mechanism to learn from exploration failures, using a structured thought management strategy to improve performance over time.

Algorithm 1: Structured Thought Management

$R = [\varnothing] * N; D = [\varnothing] * N;$
for $t \in [0, T)$:
  for $i \in [0, N)$:
    if $R[i]$ and $R[i].a' \notin D[i]$: // if has reflection
      $a_i = R[i].a'$ // action from reflection
    else: $a_i \sim \tau_\theta(a|\ldots)$ // regular planning
  if *needToReflect*: // if error happens
    $(a_j, a'_j) \sim \mathrm{REFL}_\theta(\ldots)$ // reflect
    if $R[j] \neq \varnothing$:
      $D[j].\mathrm{add}(R[j].a)$ // record wrong click
    $R[j] = (a_j, a'_j)$ // record verbal reflection
    $R[j+1:] = \varnothing; D[j+1:] = \varnothing$ // clear mem

Figure 3: An example of successful reflection trial by our zero-shot agent on MINIWOB++ task *use-autocomplete* with seed=0. Step actions are paraphrased from the actual executable ones for readability.

LI, Tao, LI, Gang, DENG, Zhiwei, et al. A Zero-Shot Language Agent for Computer Control with Structured Reflection. EMNLP, 2023.
code, link

Generalization to New Sequential Decision Making Tasks with In-Context Learning [Raparthy+ (Meta), arxiv preprint23]

- **Objective** : The goal is to enhance transformers for new sequential decision-making tasks using limited demonstrations, addressing limitations in traditional transformers.
- **Methodology**: The approach involves training on sequences of trajectories, emphasizing large model and dataset sizes, task diversity, stochastic environments, and trajectory burstiness. This includes using a causal transformer model trained on multiple trajectory sequences.
- **In-Context Learning:** The study evaluates the improved in-context learning ability on unseen tasks in diverse environments like MiniHack and Procgen, demonstrating enhanced performance in adapting to new sequential decision-making challenges.

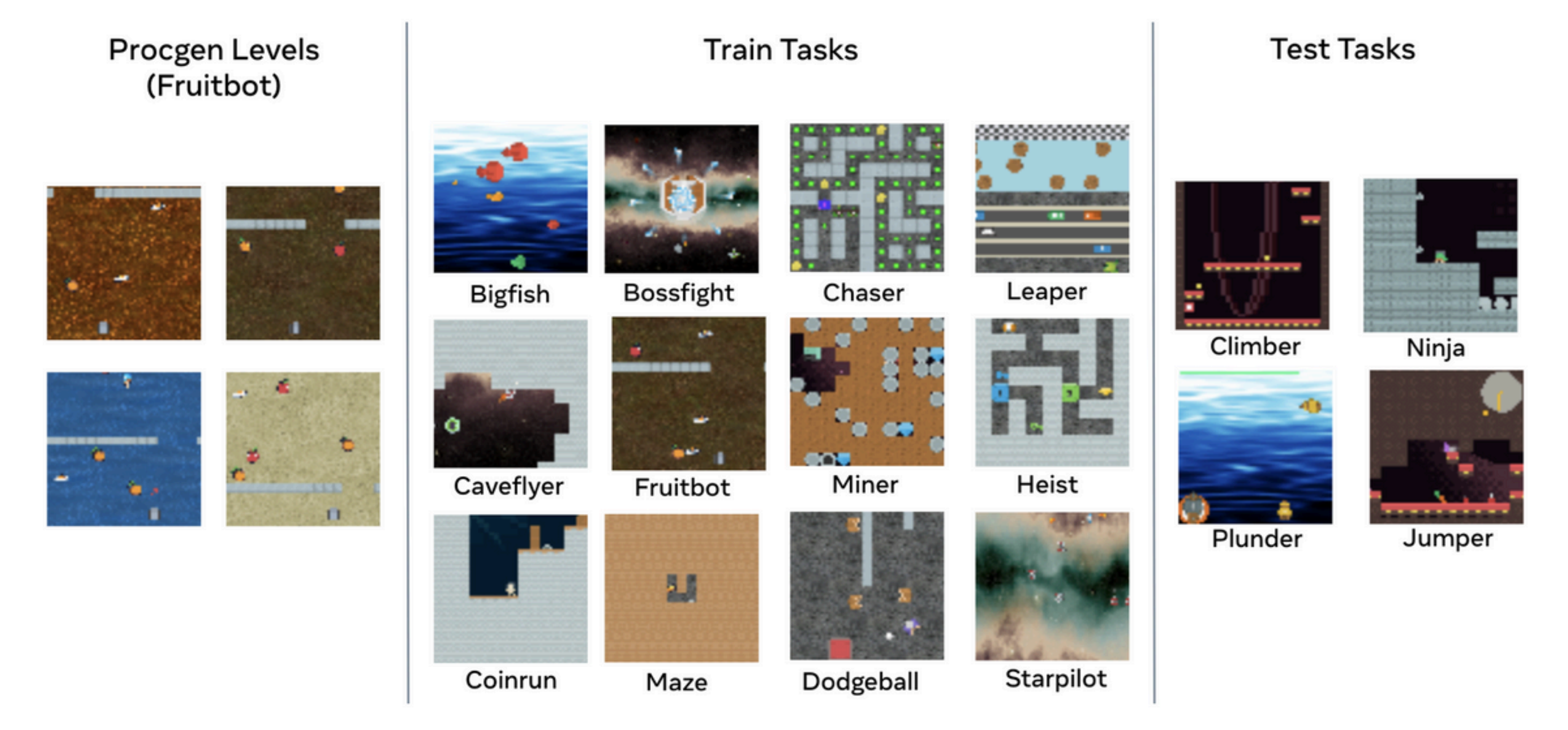


Figure 1 Illustration of Train and Test Tasks. (*Left*) A collection of procedurally generated Procgen levels from the Fruitbot task, demonstrating the complexity and diversity inherent in the environment's design. (*Middle*) Tasks used for training. (*Right*) Tasks used for testing. Note that the test tasks are entirely distinct from the training tasks, and each of them is procedurally generated, consisting of multiple levels.

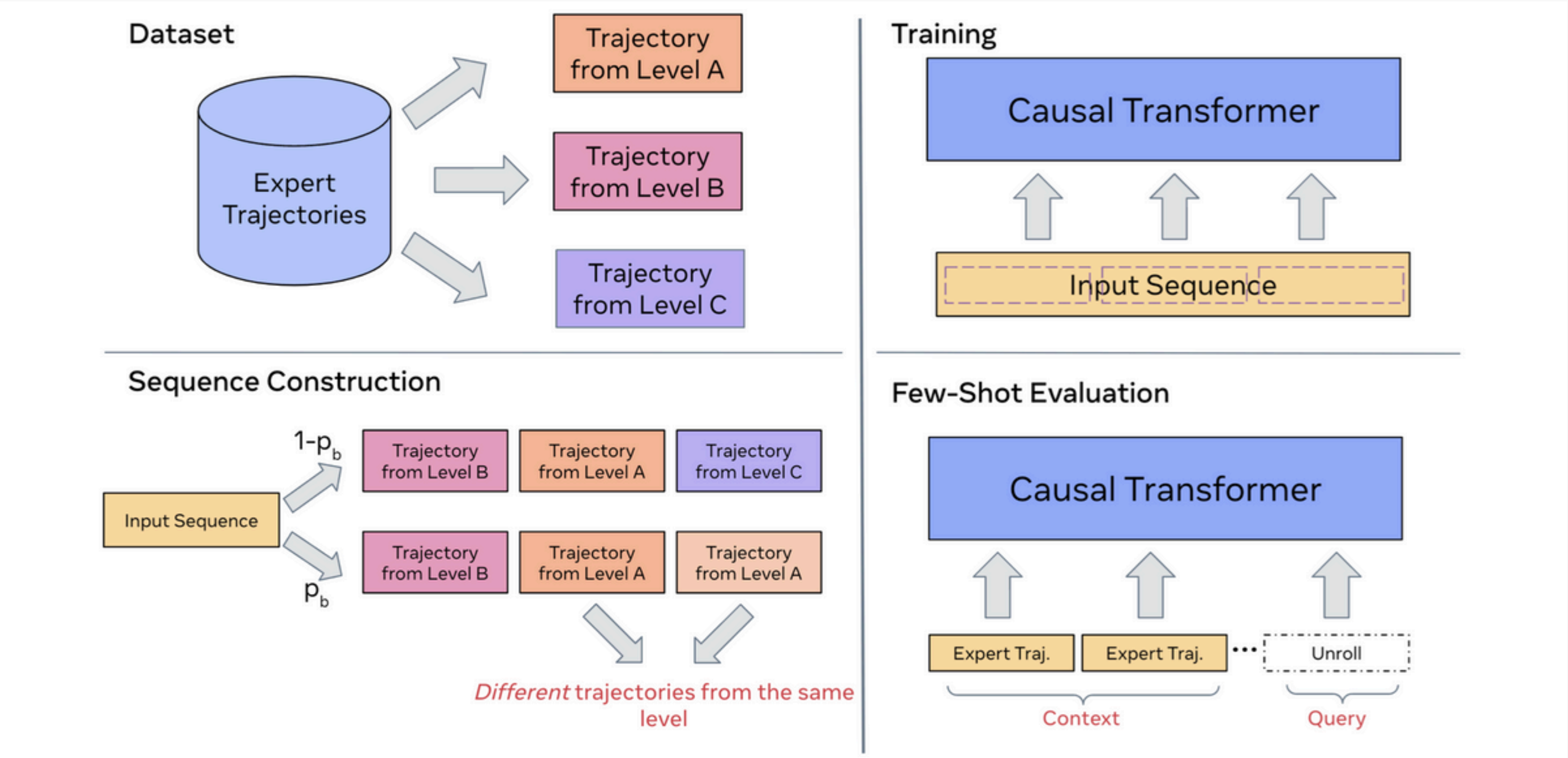


Figure 2 Experimental Setup: We create a dataset of expert trajectories by rolling out expert policies on N tasks. Given these expert trajectories, we construct multi-trajectory sequences with trajectory burstiness p_b. A sequence is bursty when there are at least two trajectories in the sequence from the same level. However, note that these trajectories are typically different due to the environment's stochasticity. These multi-trajectory sequences then serve as input to the causal transformer, which we train to predict actions. During evaluation, we condition the transformer on a few expert trajectories from an unseen task, then rollout the transformer policy until the episode terminates.

Supervised Pretraining Can Learn In-Context Reinforcement Learning [Lee+ (Stanford Univ.), arxiv preprint23]

- **Objective**: The paper aims to explore the capability of transformers in in-context reinforcement learning (RL), focusing on decision-making in bandits and Markov decision processes.
- **Methodology**: Introduces the Decision-Pretrained Transformer (DPT), a supervised pretraining method for transformers, where they predict optimal actions based on a state and in-context dataset of diverse task interactions.
- **In-Context Learning**: Demonstrates DPT's ability to handle unseen reward distributions, generalizing to new offline and online decision-making problems, and improving upon its pretraining data by exploiting latent structures. Theoretically, DPT aligns with Bayesian posterior sampling, a sample-efficient RL algorithm.

Algorithm 1 Decision-Pretrained Transformer (DPT): Training and Deployment

// Collecting pretraining dataset
Initialize empty pretraining dataset $\mathcal{B}$
for i in $[N]$ **do**
  Sample task $\tau \sim \mathcal{T}_{\text{pre}}$, in-context dataset $D \sim \mathcal{D}_{\text{pre}}(\cdot;\tau)$, query state $s_{\text{query}} \sim \mathcal{D}_{\text{query}}$
  Sample label $a^\star \sim \pi^\star_\tau(\cdot|s_{\text{query}})$ and add $(s_{\text{query}}, D, a^\star)$ to $\mathcal{B}$
end for
// Pretraining model on dataset
Initialize model M_θ with parameters θ
while not converged **do**
  Sample $(s_{\text{query}}, D, a^\star)$ from $\mathcal{B}$ and predict $\hat{p}_j(\cdot) = M_\theta(\cdot|s_{\text{query}}, D_j)$ for all $j \in [n]$
  Compute loss in (2) with respect to $a^\star$ and backpropagate to update θ.
end while
// Offline test-time deployment
Sample unknown task $\tau \sim \mathcal{T}_{\text{test}}$, sample dataset $D \sim \mathcal{D}_{\text{test}}(\cdot;\tau)$
Deploy M_θ in τ by choosing $a_h \in \text{argmax}_{a\in\mathcal{A}} M_\theta(a|s_h, D)$ at step h
// Online test-time deployment
Sample unknown task $\tau \sim \mathcal{T}_{\text{test}}$ and initialize empty $D = \{\}$
for ep in max_eps **do**
  Deploy M_θ by sampling $a_h \sim M_\theta(\cdot|s_h, D)$ at step h
  Add $(s_1, a_1, r_1, \ldots)$ to D
end for

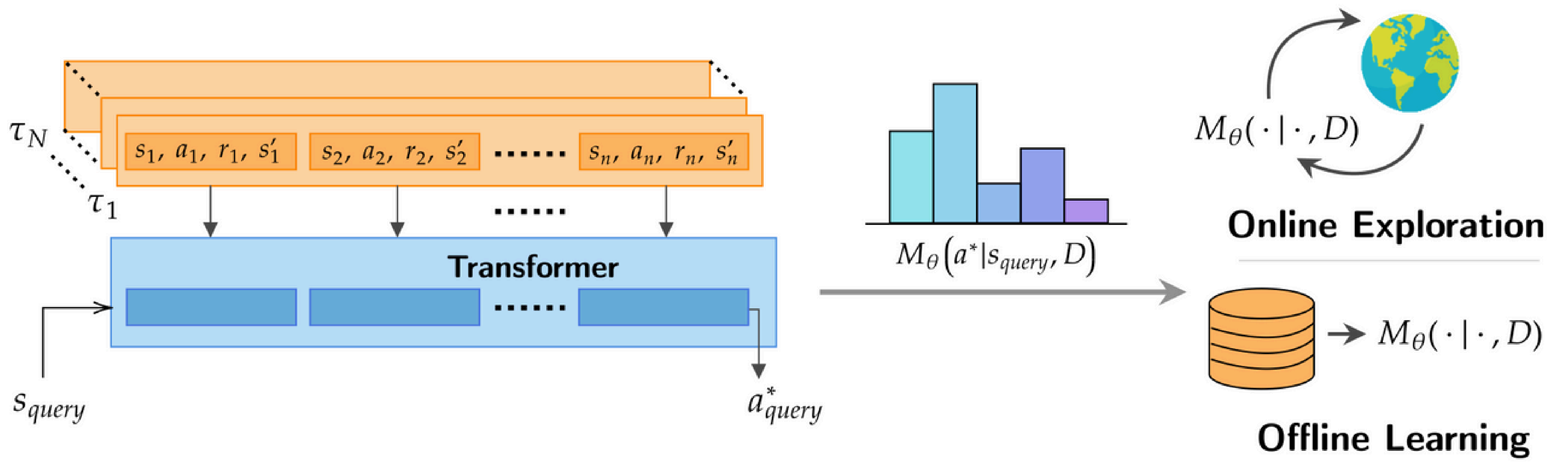


Figure 1: A transformer model M_θ is pretrained to predict an optimal action $a^\star_{\text{query}}$ from a state s_{query} in a task, given a dataset of interactions from that task. The resulting Decision-Pretrained Transformer (DPT) learns a distribution over the optimal action conditioned on an in-context dataset. M_θ can be deployed in *new* tasks online by collecting data on the fly, or offline by immediately conditioning on a static dataset.

Pearl: A Production-Ready Reinforcement Learning Agent [Zhu+ (Meta), arxiv preprint23]

- **Objective**: Develop a versatile, production-ready RL agent addressing challenges like exploration-exploitation balance, partial observability, and safety in decision-making.
- **Methodology**: Pearl features modular components, including policy learners, exploration and safety modules, history summarization, and replay buffers. It supports offline and online learning, dynamic action spaces, and integrates recent algorithmic advancements.
- **Application & Benchmarks**: Demonstrates Pearl's effectiveness through various benchmarks, including classic RL tasks, neural contextual bandits, and agent versatility tests. Highlights Pearl's adoption in industry applications like recommender systems and auction bidding.

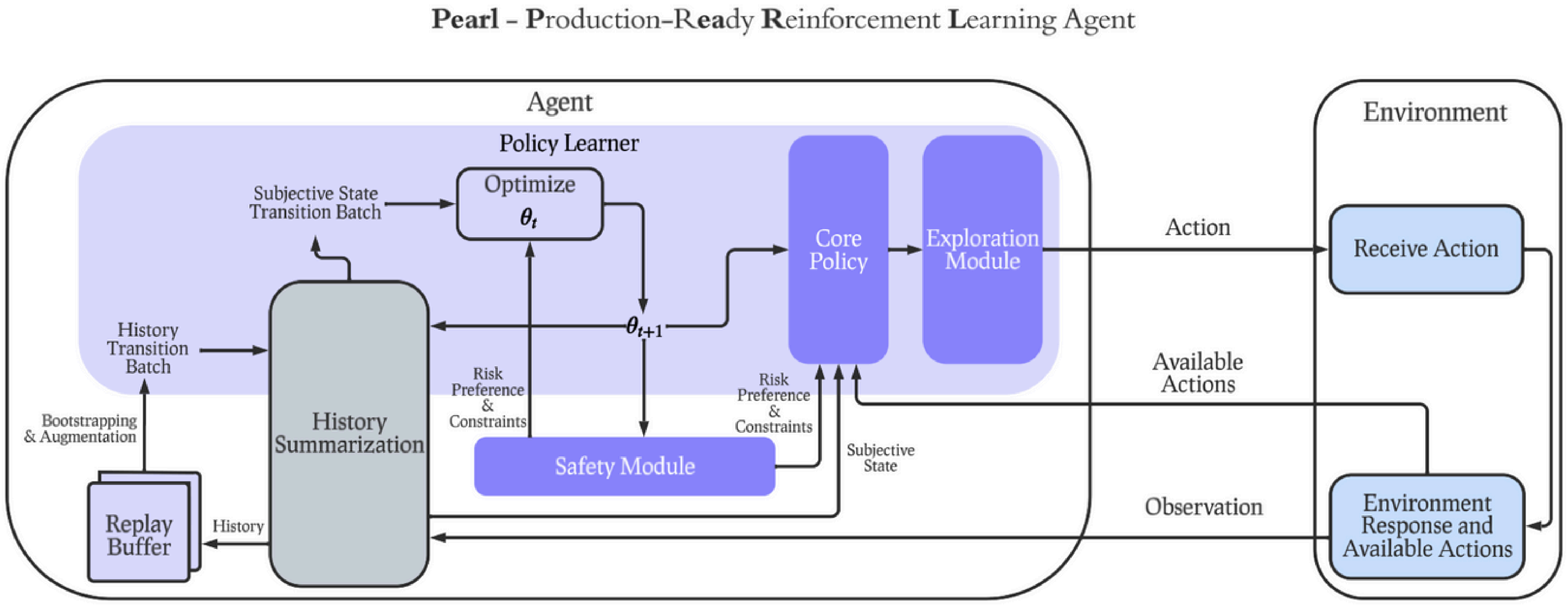


Figure 1: Pearl Agent Interface

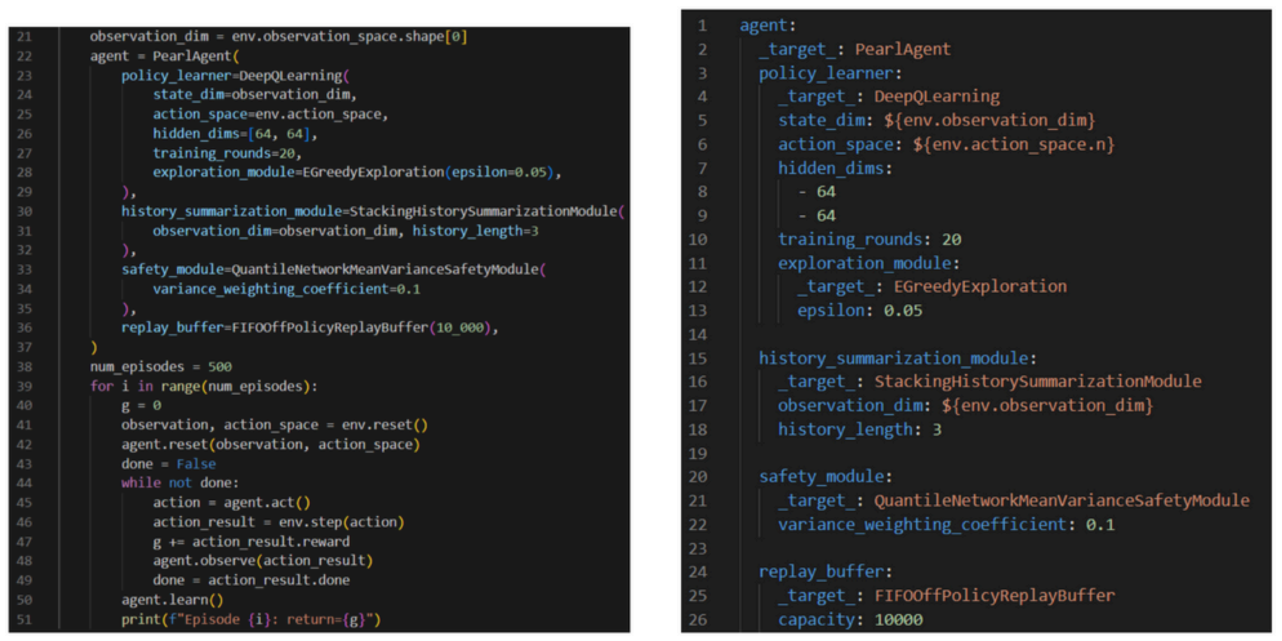

(a) PearlAgent Episodic Environment Interaction (b) Hydra Configuration for a PearlAgent

Figure 2: PearlAgent Interaction Interface and Hydra Configeration

Decision Transformer: Reinforcement Learning via Sequence Modeling [Chen+ (UC Berkeley), NeurIPS21]

- **Objective**: To reimagine RL as a sequence modeling problem, leveraging the simplicity and scalability of transformers for generative trajectory modeling.
- **Methodology**: The paper presents the Decision Transformer, an autoregressive model that, unlike traditional RL approaches, generates optimal actions conditioned on desired returns, states, and past actions.
- **Novel Approach in RL**: This framework shifts from conventional RL techniques, avoiding the need for value functions or policy gradients, and instead models a wide distribution of behaviors. It shows promising performance in various tasks, including Atari, OpenAI Gym, and Key-to-Door tasks, especially in sparse reward settings and long-term credit assignment.

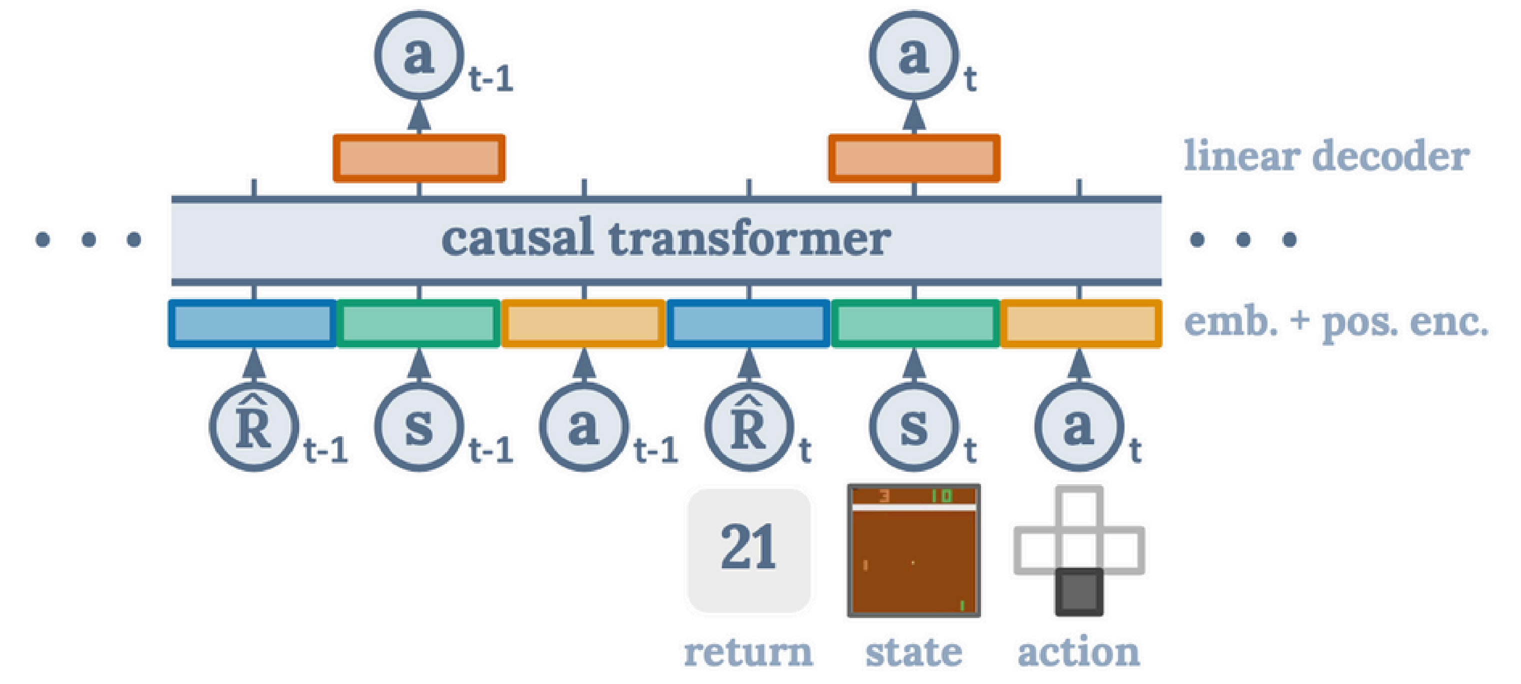


Figure 1: Decision Transformer architecture[1]. States, actions, and returns are fed into modality-specific linear embeddings and a positional episodic timestep encoding is added. Tokens are fed into a GPT architecture which predicts actions autoregressively using a causal self-attention mask.

Algorithm 1 Decision Transformer Pseudocode (for continuous actions)

```
# R, s, a, t: returns-to-go, states, actions, or timesteps
# transformer: transformer with causal masking (GPT)
# embed_s, embed_a, embed_R: linear embedding layers
# embed_t: learned episode positional embedding
# pred_a: linear action prediction layer

# main model
def DecisionTransformer(R, s, a, t):
    # compute embeddings for tokens
    pos_embedding = embed_t(t)  # per-timestep (note: not per-token)
    s_embedding = embed_s(s) + pos_embedding
    a_embedding = embed_a(a) + pos_embedding
    R_embedding = embed_R(R) + pos_embedding

    # interleave tokens as (R_1, s_1, a_1, ..., R_K, s_K)
    input_embeds = stack(R_embedding, s_embedding, a_embedding)

    # use transformer to get hidden states
    hidden_states = transformer(input_embeds=input_embeds)

    # select hidden states for action prediction tokens
    a_hidden = unstack(hidden_states).actions

    # predict action
    return pred_a(a_hidden)

# training loop
for (R, s, a, t) in dataloader:  # dims: (batch_size, K, dim)
    a_preds = DecisionTransformer(R, s, a, t)
    loss = mean((a_preds - a)**2)  # L2 loss for continuous actions
    optimizer.zero_grad(); loss.backward(); optimizer.step()

# evaluation loop
target_return = 1  # for instance, expert-level return
R, s, a, t, done = [target_return], [env.reset()], [], [1], False
while not done:  # autoregressive generation/sampling
    # sample next action
    action = DecisionTransformer(R, s, a, t)[-1]  # for cts actions
    new_s, r, done, _ = env.step(action)

    # append new tokens to sequence
    R = R + [R[-1] - r]  # decrement returns-to-go with reward
    s, a, t = s + [new_s], a + [action], t + [len(R)]
    R, s, a, t = R[-K:], ...  # only keep context length of K
```


CHEN, Lili, LU, Kevin, RAJESWARAN, Aravind, et al. Decision transformer: Reinforcement learning via sequence modeling. Advances in neural information processing systems, 2021, vol. 34, p. 15084-15097. link, code

Rainbow: Combining Improvements in Deep Reinforcement Learning [Hessel+ (DeepMind), AAAI18]

- **Objective**: Investigate the compatibility and combined effectiveness of various DQN enhancements in reinforcement learning.
- **Methodology**: Combines six key DQN improvements: Double Q-learning, Prioritized Experience Replay, Dueling Networks, Multi-step Learning, Distributional Q-Learning, and Noisy Nets, tested on the Atari 2600 suite.
- **Integrated Approach Analysis**: Presents a comprehensive analysis of how each extension contributes to overall performance, showcasing Rainbow's superiority in data efficiency and performance on Atari 2600 games compared to standalone DQN enhancements

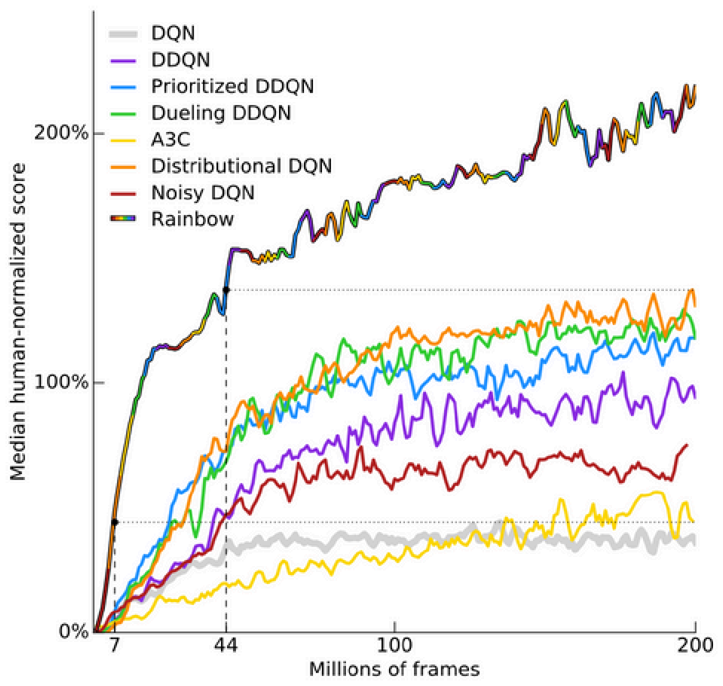


Figure 1: **Median human-normalized performance** across 57 Atari games. We compare our integrated agent (rainbow-colored) to DQN (grey) and six published baselines. Note that we match DQN's best performance after 7M frames, surpass any baseline within 44M frames, and reach substantially improved final performance. Curves are smoothed with a moving average over 5 points.

Distributional RL. We can learn to approximate the distribution of returns instead of the expected return. Recently Bellemare, Dabney, and Munos (2017) proposed to model such distributions with probability masses placed on a discrete support $\boldsymbol{z}$, where $\boldsymbol{z}$ is a vector with $N_{\text{atoms}} \in \mathbb{N}^+$ *atoms*, defined by $z^i = v_{\min} + (i-1)\frac{v_{\max}-v_{\min}}{N_{\text{atoms}}-1}$ for $i \in \{1, \ldots, N_{\text{atoms}}\}$. The approximating distribution d_t at time t is defined on this support, with the probability mass $p^i_\theta(S_t, A_t)$ on each atom i, such that $d_t = (\boldsymbol{z}, \boldsymbol{p}_\theta(S_t, A_t))$. The goal is to update θ such that this distribution closely matches the actual distribution of returns.

Dueling networks. The dueling network is a neural network architecture designed for value based RL. It features two streams of computation, the value and advantage streams, sharing a convolutional encoder, and merged by a special aggregator (Wang et al. 2016). This corresponds to the following factorization of action values:

$$q_\theta(s,a) = v_\eta(f_\xi(s)) + a_\psi(f_\xi(s), a) - \frac{\sum_{a'} a_\psi(f_\xi(s), a')}{N_{\text{actions}}},$$

where ξ, η, and ψ are, respectively, the parameters of the shared encoder f_ξ, of the value stream v_η, and of the advantage stream a_ψ; and $\theta = \{\xi, \eta, \psi\}$ is their concatenation.

Double Q-learning. Conventional Q-learning is affected by an overestimation bias, due to the maximization step in Equation 1, and this can harm learning. Double Q-learning (van Hasselt 2010), addresses this overestimation by decoupling, in the maximization performed for the bootstrap target, the selection of the action from its evaluation. It is possible to effectively combine this with DQN (van Hasselt, Guez, and Silver 2016), using the loss

$$(R_{t+1} + \gamma_{t+1} q_{\bar{\theta}}(S_{t+1}, \underset{a'}{\operatorname{argmax}}\, q_\theta(S_{t+1}, a')) - q_\theta(S_t, A_t))^2.$$

This change was shown to reduce harmful overestimations that were present for DQN, thereby improving performance.

Noisy Nets. The limitations of exploring using ϵ-greedy policies are clear in games such as Montezuma's Revenge, where many actions must be executed to collect the first reward. Noisy Nets (Fortunato et al. 2017) propose a noisy linear layer that combines a deterministic and noisy stream,

$$\boldsymbol{y} = (\boldsymbol{b} + \mathbf{W}\boldsymbol{x}) + (\boldsymbol{b}_{noisy} \odot \epsilon^b + (\mathbf{W}_{noisy} \odot \epsilon^w)\boldsymbol{x}), \quad (4)$$

where ϵ^b and ϵ^w are random variables, and $\odot$ denotes the element-wise product. This transformation can then be used in place of the standard linear $\boldsymbol{y} = \boldsymbol{b} + \mathbf{W}\boldsymbol{x}$. Over time, the network can learn to ignore the noisy stream, but will do so at different rates in different parts of the state space, allowing state-conditional exploration with a form of self-annealing.

Multi-step learning. Q-learning accumulates a single reward and then uses the greedy action at the next step to bootstrap. Alternatively, forward-view *multi-step* targets can be used (Sutton 1988). We define the truncated n-step return from a given state S_t as

$$R_t^{(n)} \equiv \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}. \quad (2)$$

A multi-step variant of DQN is then defined by minimizing the alternative loss,

$$(R_t^{(n)} + \gamma_t^{(n)} \max_{a'} q_{\bar{\theta}}(S_{t+n}, a') - q_\theta(S_t, A_t))^2.$$

Multi-step targets with suitably tuned n often lead to faster learning (Sutton and Barto 1998).

Prioritized replay. DQN samples uniformly from the replay buffer. Ideally, we want to sample more frequently those transitions from which there is much to learn. As a proxy for learning potential, prioritized experience replay (Schaul et al. 2015) samples transitions with probability p_t relative to the last encountered absolute *TD error*:

$$p_t \propto \left| R_{t+1} + \gamma_{t+1} \max_{a'} q_{\bar{\theta}}(S_{t+1}, a') - q_\theta(S_t, A_t) \right|^\omega,$$

HESSEL, Matteo, MODAYIL, Joseph, VAN HASSELT, Hado, et al. Rainbow: Combining improvements in deep reinforcement learning. In : Proceedings of the AAAI conference on artificial intelligence. 2018. [link](), [code]()

Vision-Language Models as a Source of Rewards [Baumli+ (Google DeepMind) , arxiv preprint23]

- **Objective:** Investigate the feasibility of off-the-shelf VLMs in deriving rewards for RL, enhancing agent performance in visual and language-based tasks.
- **Methodology:** Employs pretrained CLIP models to create text-based reward functions, focusing on achieving language-based goals in visual environments like Playhouse and AndroidEnv.
- **Innovation in Reward Generation:** Demonstrates how VLMs can train RL agents without environment-specific finetuning, showing that larger VLMs lead to more accurate rewards and more capable agents.

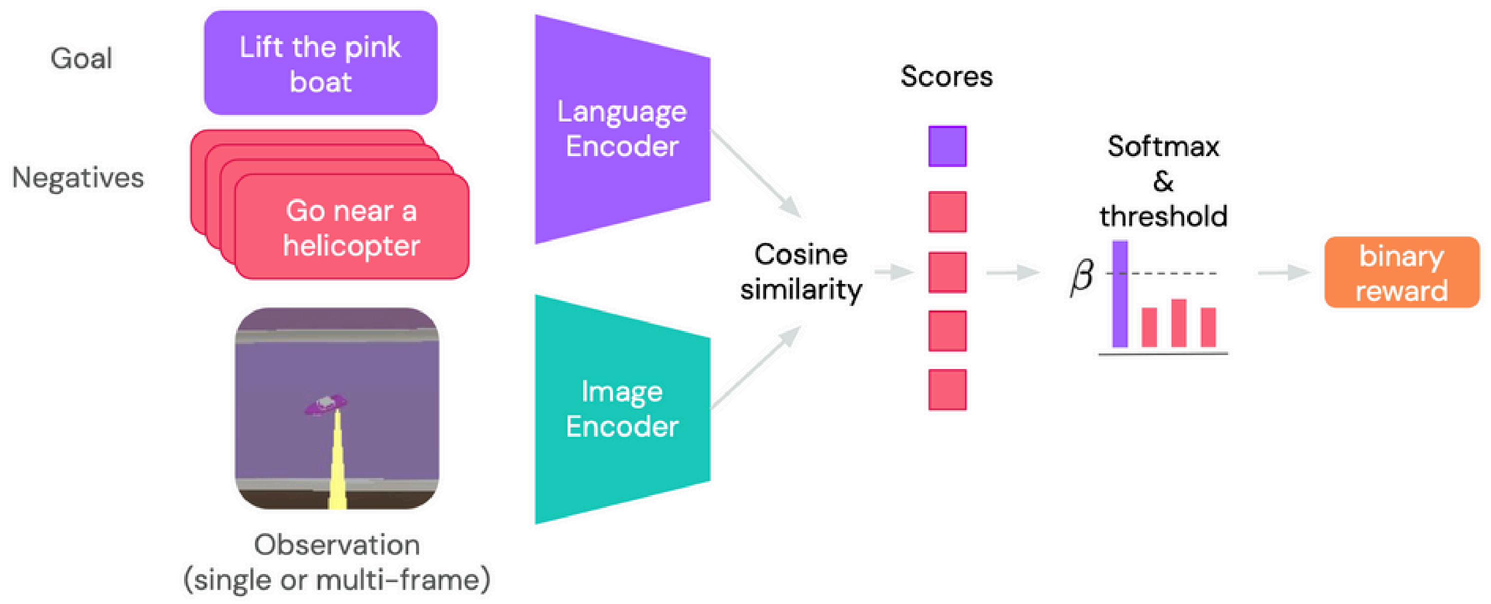


Figure 1: Architecture for Vision-Language Models (VLMs) as rewards. The VLM trained contrastively contains an image encoder f_θ and language encoder g_θ. We embed the current environment observation frame(s) using the image encoder, along with the desired goal language descriptions l and negative language descriptions using the language encoder. The reward is computed by taking the cosine similarity scores and applying softmax and thresholding.

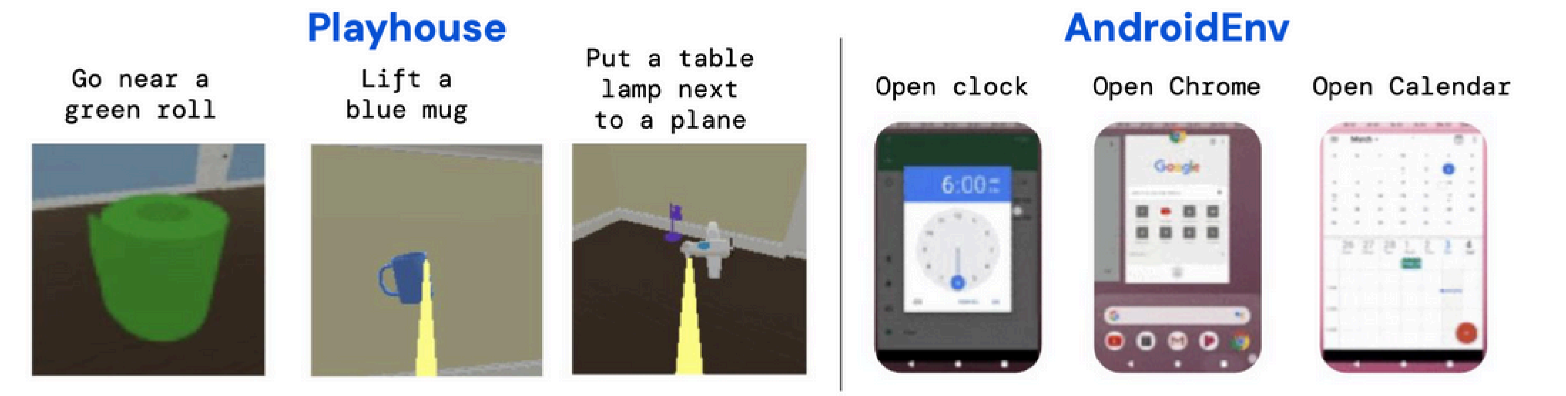


Figure 2: Environments and example tasks. (**Left**) Playhouse [27] consists of *Find*, *Lift*, and *Pick and Place* tasks. (**Right**) AndroidEnv [29] consists of opening app tasks across various apps on Android.

BAUMLI et al. Vision-Language Models as a Source of Rewards.
arXiv preprint arXiv2312.09187, 2023. link.

Do As I Can, Not As I Say: Grounding Language in Robotic Affordances [Ahn+ (Google), arxiv preprint22]

- **Objective**: Develop a method to utilize language models for guiding robotic systems in executing complex tasks based on natural language instructions.
- **Methodology**: Combines the predictive capabilities of large language models (PaLM) with the practical affordances of robotic systems. The language model generates task sequences, while the robotic system executes them based on its capabilities and environment understanding.
- **Novel Integration for Robotic Tasks**: Showcases how combining language understanding with robotic action enables the execution of complex, multi-step tasks in real-world settings. Demonstrates this integration's effectiveness in enhancing task planning and execution by a robotic system.

Algorithm 1 SayCan

Given: A high level instruction i, state s_0, and a set of skills Π and their language descriptions ℓ_Π

$n = 0, \pi = \emptyset$
while $\ell_{\pi_{n-1}} \neq$ "done" **do**
  $\mathcal{C} = \emptyset$
  for $\pi \in \Pi$ and $\ell_\pi \in \ell_\Pi$ **do**
    $p_\pi^{\text{LLM}} = p(\ell_\pi | i, \ell_{\pi_{n-1}}, ..., \ell_{\pi_0})$ ▷ Evaluate scoring of LLM
    $p_\pi^{\text{affordance}} = p(c_\pi | s_n, \ell_\pi)$ ▷ Evaluate affordance function
    $p_\pi^{\text{combined}} = p_\pi^{\text{affordance}} p_\pi^{\text{LLM}}$
    $\mathcal{C} = \mathcal{C} \cup p_\pi^{\text{combined}}$
  end for
  $\pi_n = \arg\max_{\pi \in \Pi} \mathcal{C}$
  Execute $\pi_n(s_n)$ in the environment, updating state s_{n+1}
  $n = n + 1$
end while

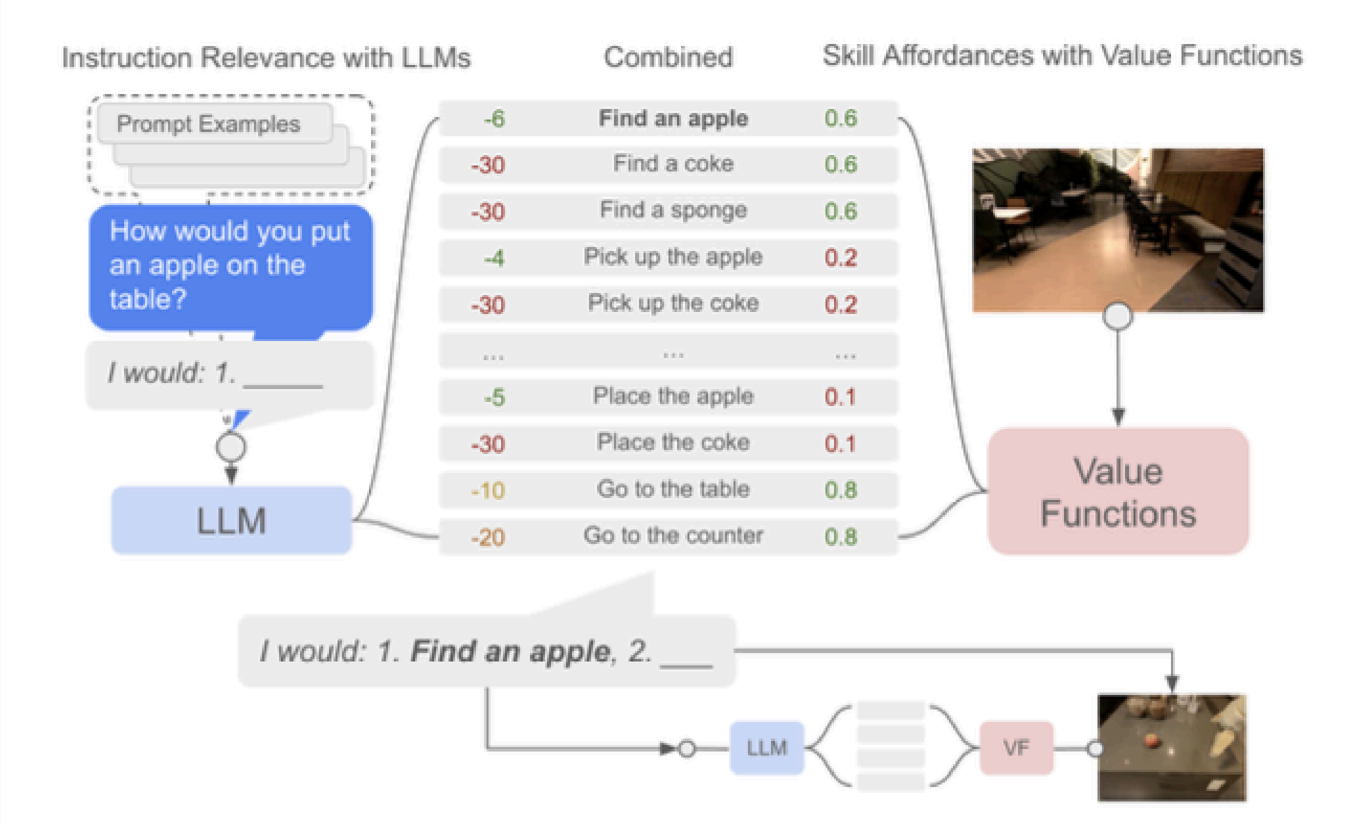


Figure 3: Given a high-level instruction, SayCan combines probabilities from a LLM (the probability that a skill is useful for the instruction) with the probabilities from a value function (the probability of successfully executing said skill) to select the skill to perform. This emits a skill that is both possible and useful. The process is repeated by appending the skill to the response and querying the models again, until the output step is to terminate. Appendix Figures 12 and 2 focus on the LLM and VFS components.

Figure 1: LLMs have not interacted with their environment and observed the outcome of their responses, and thus are not grounded in the world. SayCan grounds LLMs via value functions of pretrained skills, allowing them to execute real-world, abstract, long-horizon commands on robots.

AHN, Michael, BROHAN, Anthony, BROWN, Noah, et al. Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691, 2022. link, code

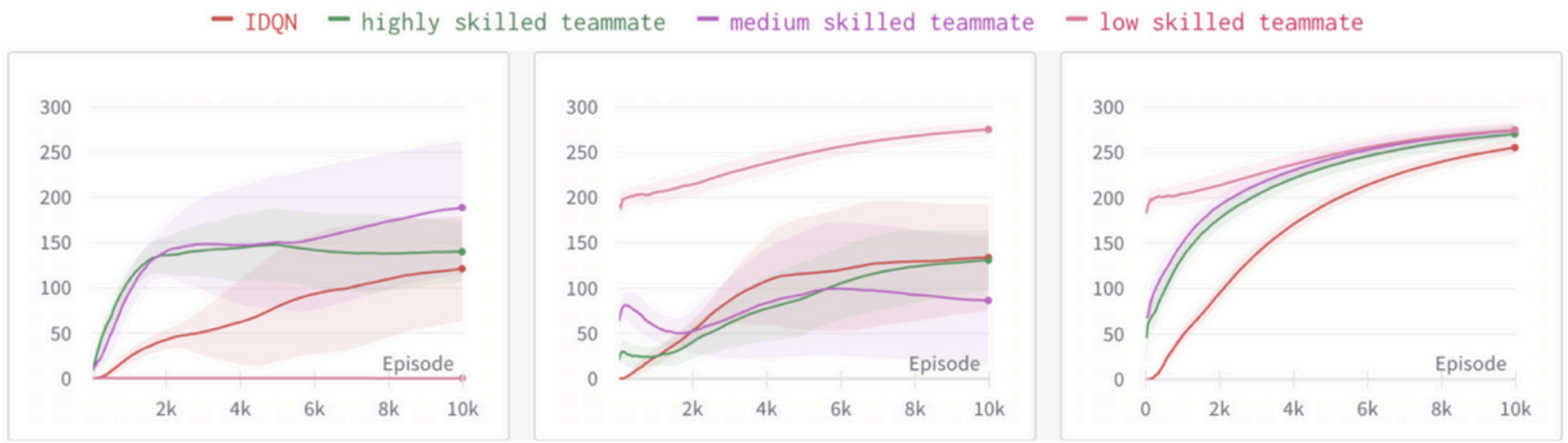
Curriculum Learning for Cooperation in Multi-Agent Reinforcement Learning [Bhati+ (AI Redefined), NeurIPS23]

- **Objective**: The study aims to explore and establish efficient training strategies for cooperative multi-agent systems, focusing on the type of teammates and the sequence of their skill levels during training.
- **Methodology**: The approach involves creating a population of teammate agents with varying skill levels and training a novice agent (the student) with these teammates. It uses the game "Overcooked" as the test environment and examines the impact of different teammate skill levels and curricula on the student agent's learning and team performance.
- **Innovative Approach in Cooperative Learning**: The paper investigates the effects of training with teammates of different skill levels and curricula (increasing vs. decreasing skill levels) on the student agent's learning and overall team performance. It finds that a curriculum of teammates with decreasing skill levels often leads to better team rewards and learning outcomes for the student agent, compared to increasing skill level curricula or training with a single pre-trained teammate.

Train with pre-trained agents To determine which teammate is a good teammate, we pair a new student agent that learns from scratch with a pre-trained (non-learning) teammate that is sampled from the population of teammates $\mathfrak{B}$. We specifically use three teammates $\{\pi'_{t_1}, \pi'_{t_2}, \pi'_{t_3}\}$ where $t_1 < t_2 < t_3$. We call the π'_{t_1} as the less trained (or less skilled) agent, the π'_{t_2} as the medium trained agent, and the π'_{t_3} as the highly trained agent.

Train with a curriculum of pre-trained agents In order to train with a curriculum of pre-trained agents, we can make several combinations of the three pre-trained agents in $\mathfrak{B}$ (i.e., low-skilled, medium-skilled, and highly-skilled). We divide the training procedure of the student agent into three equal parts of the number of episodes. For the total number of episodes is K, we pair the student agent with a different agent for every $K/3$ episodes. We create two curricula of teammates:

1. **Increasing curriculum**: We increase the skill level of the teammates during training. Therefore, during the first $K/3$ episodes, the student agent is trained alongside the low-skilled teammate, then for the next $K/3$ episodes, it is trained alongside the medium-skilled teammate, and for the last $K/3$ episodes, it is trained alongside the highly-skilled teammate.
2. **Decreasing curriculum**: We decrease the skill level of the teammates during training. Therefore, similar to the increasing curriculum, the student is first paired with the highly-skilled teammate, then the medium-skilled teammate, and finally the low-skilled teammate, each for $K/3$ episodes.



(a) student reward (b) teammate reward (c) total reward

Figure 3: Individual and total reward for the student and teammate agents when the student (that is trained from scratch) is paired with a non-learning teammate. All means and standard deviations are aggregated over 5 seeds.

BHATI, Rupali, GOTTIPATI, Sai Krishna, MARS, Clodéric, et al.Curriculum Learning for Cooperation in Multi-Agent Reinforcement Learning. arXiv preprint arXiv:2312.11768, 2023, **link**

Critic-Guided Decision Transformer for Offline Reinforcement Learning [Wang+ (Shangai Univ), AAAI24]

- **Objective**: The study aims to address limitations of Return-Conditioned Supervised Learning (RCSL) in offline RL, particularly in stochastic environments and stitching scenarios, by introducing the Critic-Guided Decision Transformer (CGDT).
- **Methodology**: CGDT combines the predictability of long-term returns from value-based methods with the trajectory modeling capability of Decision Transformers. It integrates a learned value function (the critic) to guide policy training, ensuring alignment between target returns and expected returns of actions.
- **Advancements in Offline RL**: The paper highlights CGDT's effectiveness in handling stochasticity and stitching problems, which are challenges for traditional RCSL. CGDT's integration of a value function bridges the deterministic nature of RCSL with the probabilistic characteristics of value-based methods, advancing state-of-the-art in offline RL and expanding RCSL's applicability in various RL tasks.

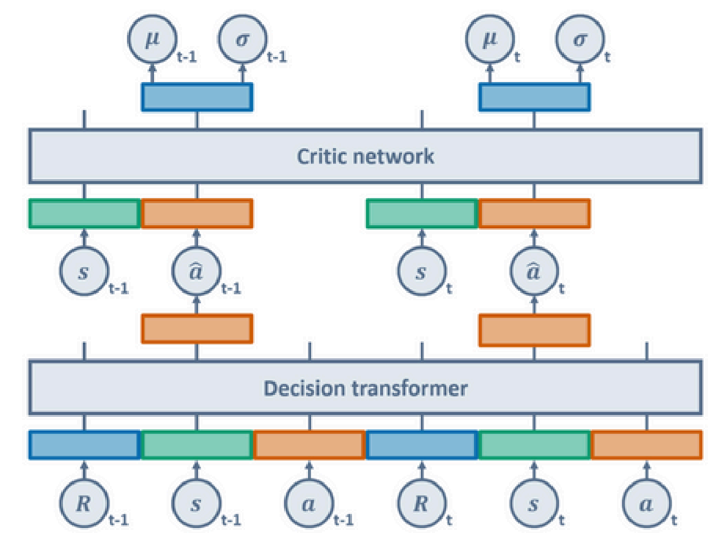


Figure 1: *Critic-Guided Decision Transformer framework.* The lower part is a vanilla Decision Transformer that takes the states s, actions a, and target returns R as inputs to predict the next action $\hat{a}_t$ for each state s_t. The predicted actions are then passed through a critic, which is a Gaussian distribution with expected return mean μ_t and variance σ_t learned from offline data. By minimizing the distance between the expected returns of the predicted actions and the target returns, e.g. $\|(R_t - \mu_t)/\sigma_t\|_2$, the critic guides the policy to take actions that are consistent with the target returns.

Algorithm 1: Critic-Guided Decision Transformer

Input: Offline dataset $\mathcal{D}$, critic Q_ϕ, policy π_θ, iterations M, N, asymmetric critic coefficient τ_c, expectile regression parameter τ_p, and balance weight α.

\\\\ Asymmetric Critic Training
for $i = 1, ..., M$ **do**
  Sample a batch of trajectories (s_t, a_t, r_t) from $\mathcal{D}$;
  Compute return of sub-trajectory $\tau_{t:T}$, $R_t = \sum_t^T r_t$;
  Update Q_ϕ with gradient:

$$\mathbb{E}_{(s_t,a_t,R_t)}\left[\nabla_\phi \mathcal{L}_Q(\phi)\right];$$

end for
\\\\ Critic-Guided Policy Training
$\alpha' \leftarrow 0$
for $j = 1, ..., N$ **do**
  $\alpha' \leftarrow \alpha' + \alpha/N$
  Sample a batch of trajectories (s_t, a_t, r_t) from $\mathcal{D}$;
  Compute return of sub-trajectory $\tau_{t:T}$, $R_t = \sum_t^T r_t$;
  Predict action $\hat{a}_t \sim \pi_\theta(\cdot|\tau_{0:t-1}, s_t, R_t)$;
  Predict return $(\mu_t, \sigma_t) \sim Q_\phi(\cdot|\tau_{0:t-1}, s_t, \hat{a}_t)$;
  Compute *expectile regression* loss: $\mathcal{L}_2^{\tau_p}(\frac{R_t-\mu_t}{\sigma_t})$;
  Update π_θ with gradient:

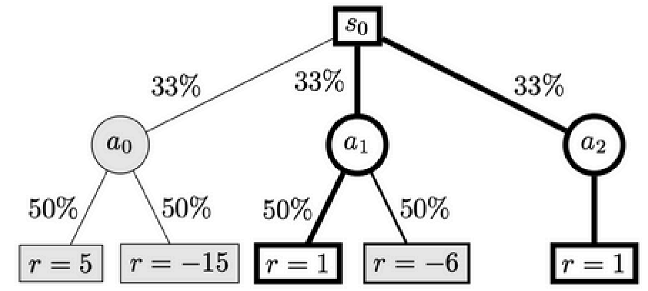
$$\mathbb{E}_{(s_t,a_t,\hat{a}_t,R_t)}\left[\nabla_\theta \mathcal{L}_2(a_t,\hat{a}_t) + \alpha' \nabla_\theta \mathcal{L}_2^{\tau_p}(\frac{R_t-\mu_t}{\sigma_t})\right];$$

end for
return π_θ

WANG, Yuanfu, YANG, Chao, WEN, Ying, et al. Critic-Guided Decision Transformer for Offline Reinforcement Learning. arXiv preprint arXiv:2312.13716, 2023. [link]

You Can’t Count on Luck: Why Decision Transformers and RvS Fail in Stochastic Environments [Paster+ (Toronto Univ.), NeurIPS22]

- **Objective**: The study aims to address the shortcomings of Reinforcement Learning via Supervised Learning (RvS) approaches, like Decision Transformers, in stochastic environments. It identifies the issue of these models failing due to their reliance on trajectory returns, which can be influenced by luck.
- **Methodology**: The paper introduces ESPER (Environment-Stochasticity-Independent Representations), a method that conditions on average cluster returns independent of environmental stochasticity. This involves using adversarial learning to cluster trajectories, ensuring that the cluster assignments do not contain information about the stochastic outcomes of the environment.
- **Innovative Solution in Stochastic RL**: The key innovation lies in ESPER's approach to learning and conditioning on environment-stochasticity-independent representations. This allows for stronger alignment between target return and expected performance in real environments, addressing the fundamental issue identified with existing RvS approaches in stochastic settings. The paper demonstrates the effectiveness of ESPER in several challenging stochastic offline RL tasks, showing significant improvements over traditional conditioning on returns.



Conditioned Return	$\mathbb{E}[r]$	Actions
$r = -15$	-5.0	$\{a_0\}$
$r = -6$	-2.5	$\{a_1\}$
$r = 1$	-0.17	$\{a_1, a_2\}$
$r = 5$	-5.0	$\{a_0\}$

Figure 1: **Left:** A simple gambling environment with three actions where return-conditioned algorithms such as Decision Transformer [1] will fail, *even with infinite data*. The optimal action, a_2, will always grant the agent 1 reward while gambling (a_0 and a_1) give a stochastic amount of reward. The numbers (33%) above each action represent the data collection policy. **Right:** The second column represents the performance of a policy behaviorally cloned from trajectories achieving the reward in the first column. For example, conditioned on the agent receiving $r = 1$, a third of trajectories take a_1 and two-thirds take a_2. Averaging the returns $-2.5 \times 1/3 + 1 \times 2/3 = -0.17$. **No matter how much data the model is trained on, when conditioned on receiving a reward of 1, it will always gamble and take a_1 some of the time, rather than just taking a_2, which guarantees the reward.**

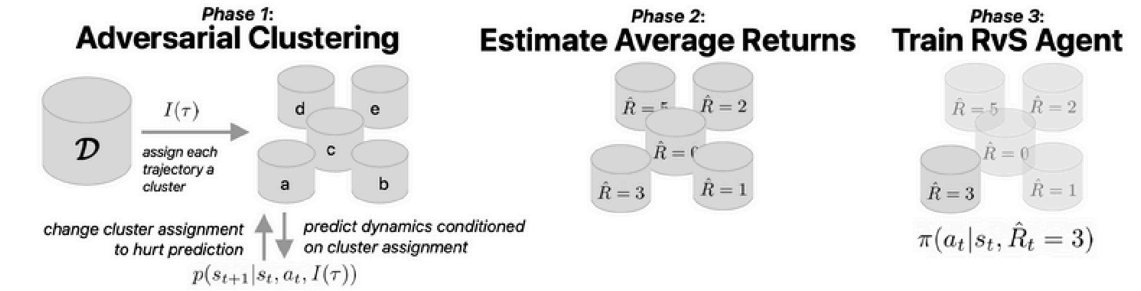


Figure 2: ESPER learns a policy that conditions on a desired *expected* return. In phase 1 of the algorithm, a function is learned using adversarial learning that assigns each trajectory in the dataset to a cluster such that the cluster assignments do not contain information about the stochastic outcomes of the environment that can help a dynamics model cheat to predict next states. In phase 2, the average return in each cluster is calculated. In phase 3, an RvS agent is trained to predict the next action given the current state and the estimated average return.

PASTER, Keiran, MCILRAITH, Sheila, et BA, Jimmy. You can’t count on luck: Why decision transformers and rvs fail in stochastic environments. Advances in Neural Information Processing Systems, 2022, vol. 35, p. 38966-38979. link

PROGPROMPT: Generating Situated Robot Task Plans using Large Language Models [Singh+ (NVIDIA), IEEE23]

- **Objective**: To enhance robot task planning by leveraging the strengths of LLMs in commonsense reasoning and code understanding. The goal is to generate executable robot task plans that are adaptable to different environments and tasks.
- **Methodology**: Introduces ProgPrompt, a prompting scheme that uses Pythonic program structures as prompts for LLMs. This method includes import statements of available actions, lists of environment objects, and sample task plans as functions. The LLM generates task plans based on these prompts, which are then executed by a robot in a simulated or physical environment.
- **Innovative Approach to Task Planning**: ProgPrompt represents a significant advancement in robot task planning, demonstrating that LLMs can be effectively used to generate detailed and executable task plans. This approach combines programming language prompts with LLMs' natural language processing capabilities, resulting in more accurate and context-aware task plans suitable for diverse robotic applications. The paper showcases the effectiveness of this method through experiments in both virtual and physical environments, highlighting its adaptability and potential for wide-ranging real-world applications.

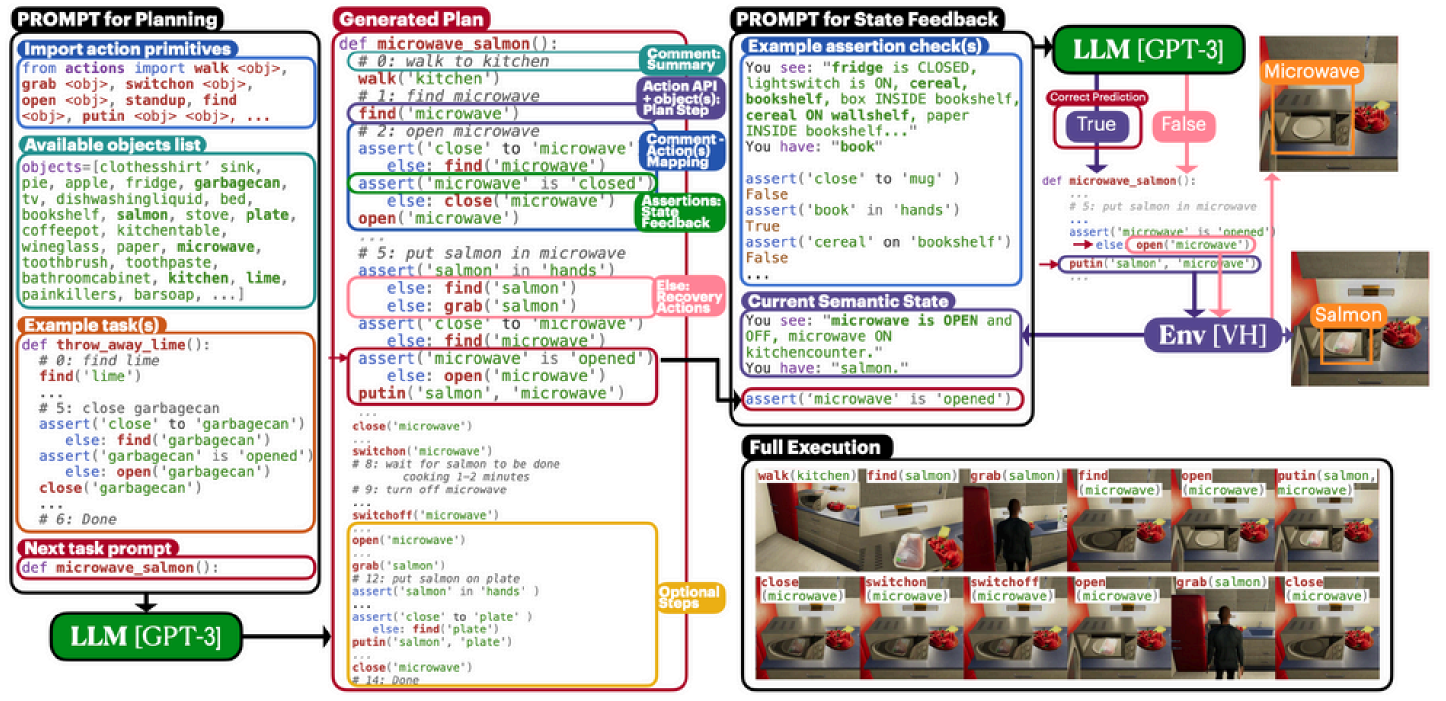


Fig. 2: Our PROGPROMPTs include `import` statement, object list, and example tasks (*PROMPT for Planning*). The *Generated Plan* is for `microwave salmon`. We highlight prompt comments, actions as imported function calls with objects as arguments, and `assertions` with recovery steps. *PROMPT for State Feedback* represents example assertion checks. We further show execution of the program. We illustrate a scenario where an assertion succeeds or fails, and how the generated plan corrects the error before executing the next step. *Full Execution* of the program is shown in bottom-right.

Training language models to follow instructions with human feedback [Ouyang+ (OpenAI), NeurIPS22]

- **Objective:** The goal is to align LMs with user intentions across a wide range of tasks by fine-tuning them with human feedback. The study aims to create models that are more helpful, honest, and harmless, aligning better with user needs and ethical guidelines.
- **Methodology:** The approach involves three key steps:
    - **Supervised Fine-Tuning (SFT)**: Collecting demonstration data to train a supervised policy. Labelers provide demonstrations of desired behavior on input prompts.
    - **Reward Modeling (RM):** Collecting comparison data to train a reward model. Labelers rank model outputs, indicating preferences, and a reward model is trained to predict human-preferred outputs.
    - **Reinforcement Learning via Proximal Policy Optimization (PPO):** Using the output of the reward model as a reward signal, the supervised policy is fine-tuned to optimize this reward.
- **Advancements in Human-Aligned Language Modeling:** This method marks a significant step in aligning LMs with human preferences. The process involves careful collection and curation of human feedback to guide the training of LMs. The paper demonstrates that the InstructGPT models, which result from this process, show improvements in following instructions, truthfulness, and reductions in toxic output generation, despite having fewer parameters compared to models like GPT-3. The approach suggests that fine-tuning with human feedback is a promising direction for creating LMs that are more aligned with human intent and ethical considerations.

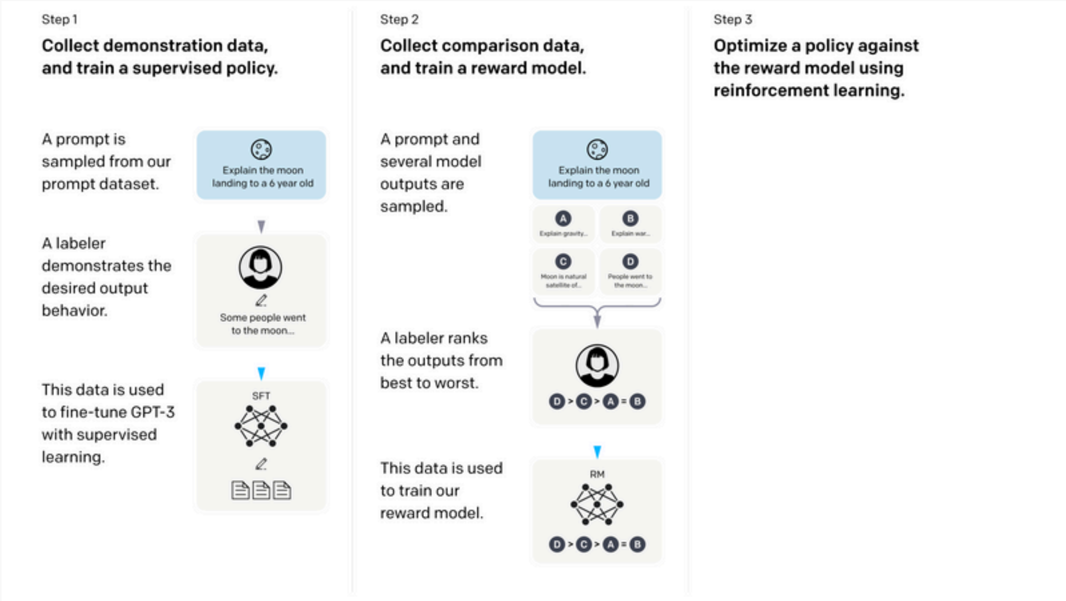


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

OUYANG, Long, WU, Jeffrey, JIANG, Xu, et al. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 2022, vol. 35, p. 27730-27744. [link]

Direct Preference Optimization: Your Language Model is Secretly a Reward Model [Rafailov+ (Stanford+), NeurIPS23]

- **Objective:** To simplify the process of training LMs to align with human preferences, bypassing complex RL-based methods. It aims to directly optimize LMs with human preferences using a new approach called Direct Preference Optimization (DPO).
- **Methodology:** DPO involves a new parameterization of the reward model in RLHF (Reinforcement Learning from Human Feedback), enabling extraction of the optimal policy in closed form. This method simplifies the training process by using a binary cross-entropy loss, avoiding the need for sampling from the LM during fine-tuning or extensive hyperparameter tuning.
- **Innovative Approach in Preference Learning:** The paper's central contribution is introducing a stable, efficient, and computationally lightweight approach to align LMs with human preferences. DPO outperforms existing RLHF methods, such as PPO, in tasks like sentiment modulation, summarization, and dialogue, while being substantially simpler to implement and train. The experiments demonstrate DPO's effectiveness in fine-tuning LMs to human preferences, offering a new paradigm in preference-based LM training.

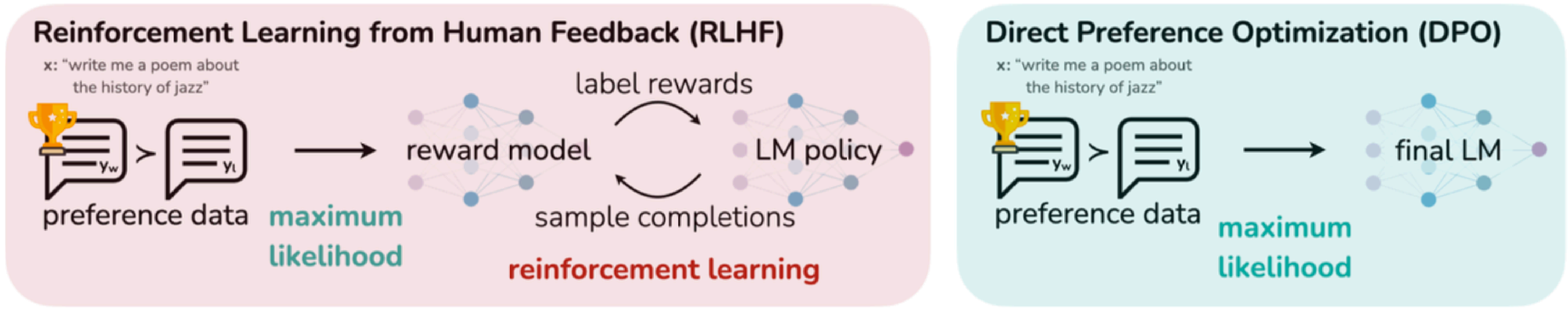


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

RAFAILOV, Rafael, SHARMA, Archit, MITCHELL, Eric, et al.Direct preference optimization: Your language model is secretly a reward model. arXiv preprint arXiv:2305.18290, 2023. link

CLIPORT: What and Where Pathways for Robotic Manipulation [Shridhar+ (Washington Univ.), PMLR22]

- **Objective**: Develop an end-to-end framework, CLIPort, for language-conditioned robotic manipulation, integrating semantic understanding and spatial precision for a variety of tabletop tasks.
- **Methodology**: CLIPort combines CLIP for semantic understanding (what) with the spatial precision of Transporter (where), in a two-stream architecture. It's trained on language-conditioned manipulation tasks, ranging from packing objects to complex actions like folding cloths.
- **Innovation in Robotic Manipulation:** The paper's key innovation lies in its ability to ground abstract language instructions into precise physical actions using a blend of semantic and spatial understanding. This approach enables robots to perform a wide range of tasks based on natural language commands, showcasing data efficiency in few-shot settings and effective generalization to both seen and unseen semantic concepts.

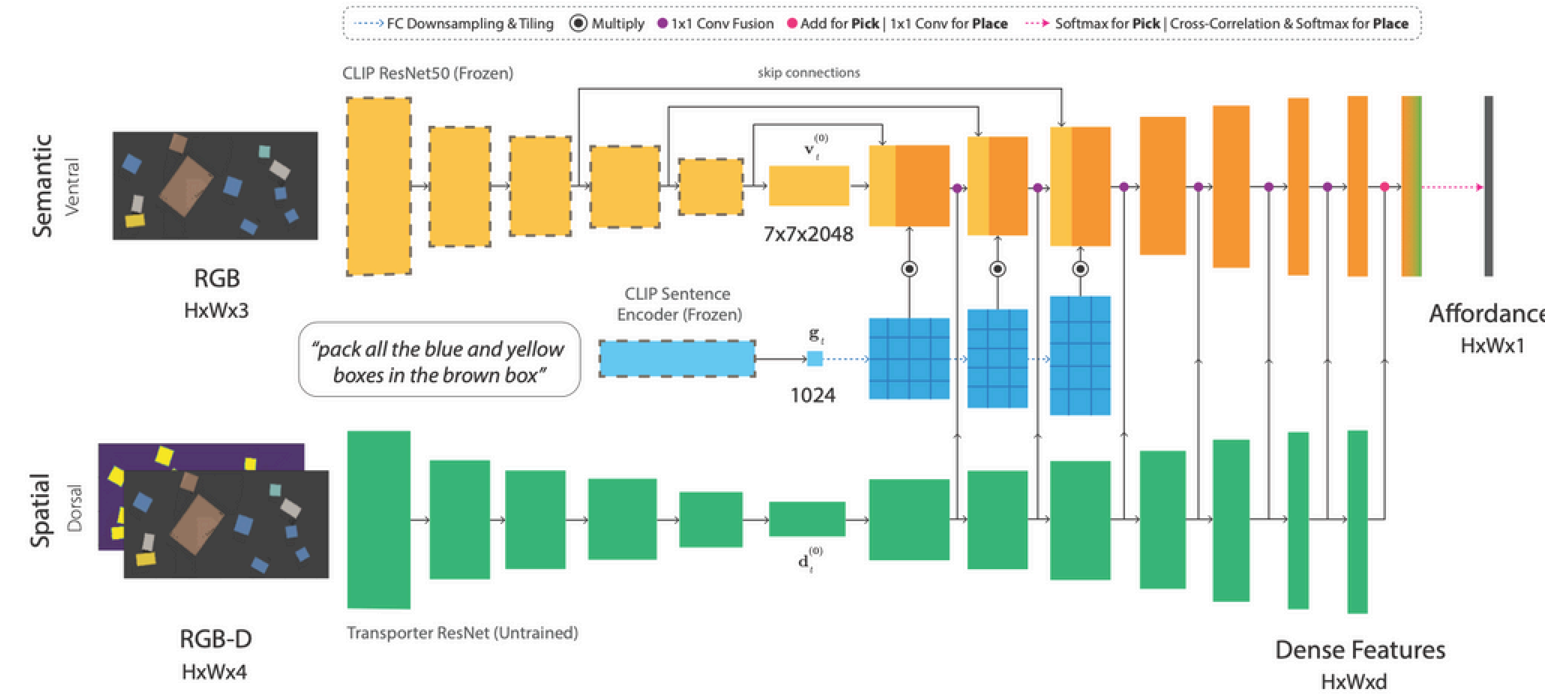


Figure 2. **CLIPort Two-Stream Architecture.** An overview of the **semantic** and **spatial** streams. The **semantic** stream uses a frozen CLIP ResNet50 [1] to encode RGB input, and its decoder layers are conditioned with tiled language features from the CLIP sentence encoder. The **spatial** stream encodes RGB-D input, and its decoder layers are laterally fused with the **semantic** stream. The final output is a map of dense pixelwise features that is used for pick or place affordance predictions. This same two-stream architecture is used in all 3 Fully-Convolutional-Networks f_{pick}, Φ_{query}, and Φ_{key} with f_{pick} is used to predict pick actions, and Φ_{query} and Φ_{key} are used to predict place actions. See Appendix C for the exact architecture.

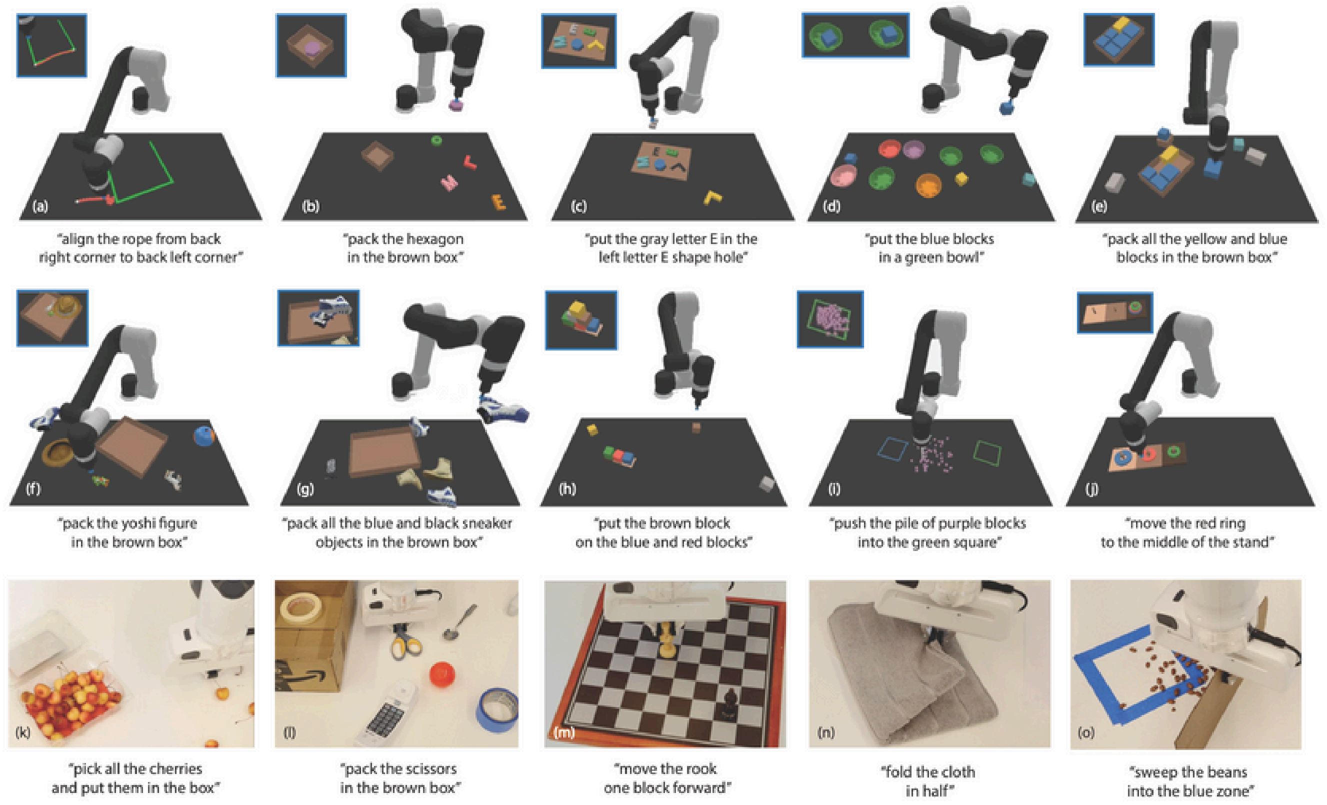


Figure 1. **Language-Conditioned Manipulation Tasks:** CLIPort is a broad framework applicable to a wide range of language-conditioned manipulation tasks in tabletop settings. We conduct large-scale experiments in Ravens [2] on 10 simulated tasks (a-j) with 1000s of unique instances per task. See Appendix A for challenges pertaining to each task. CLIPort can even learn one multi-task model for all 10 tasks that achieves better or comparable performance to single-task models. Similarly, we demonstrate our approach on a Franka Panda manipulator with one multi-task model for 9 real-world tasks (k-o; only 5 shown) trained with just 179 image-action pairs.

SHRIDHAR, Mohit, MANUELLI, Lucas, et FOX, Dieter. Cliport: What and where pathways for robotic manipulation. In : Conference on Robot Learning. PMLR, 2022. p. 894-906. link, code

VOYAGER: An Open-Ended Embodied Agent with Large Language Models [Wang+ (NVIDIA), arxiv preprint23]

- **Objective**: The study aims to develop an embodied agent capable of continuous exploration, diverse skill acquisition, and making novel discoveries in Minecraft without human intervention. VOYAGER is designed to be an LLM-powered agent that can self-improve through exploration and skill development.
- **Methodology**: VOYAGER integrates three key components:
  - **Automatic Curriculum**: Maximizes exploration by suggesting new tasks based on the agent's state and progress.
  - **Skill Library:** Stores and retrieves executable code representing complex behaviors, allowing for continual skill development.
  - **Iterative Prompting Mechanism**: Uses a novel approach for self-improvement by incorporating environment feedback, execution errors, and self-verification to refine executable programs.
- **Innovation in Embodied Agents**: VOYAGER represents a significant advancement in the field of AI and embodied agents. It successfully uses an LLM (GPT-4) to generate task proposals and action plans in the form of executable code, effectively combining NLP capabilities with embodied control. This enables VOYAGER to perform a wide array of tasks in Minecraft, demonstrating its strong in-context lifelong learning capabilities, and its ability to utilize learned skills in novel scenarios. The agent showcases significant improvements in exploration, skill acquisition, and task execution, outperforming state-of-the-art techniques in various benchmarks within the Minecraft environment.

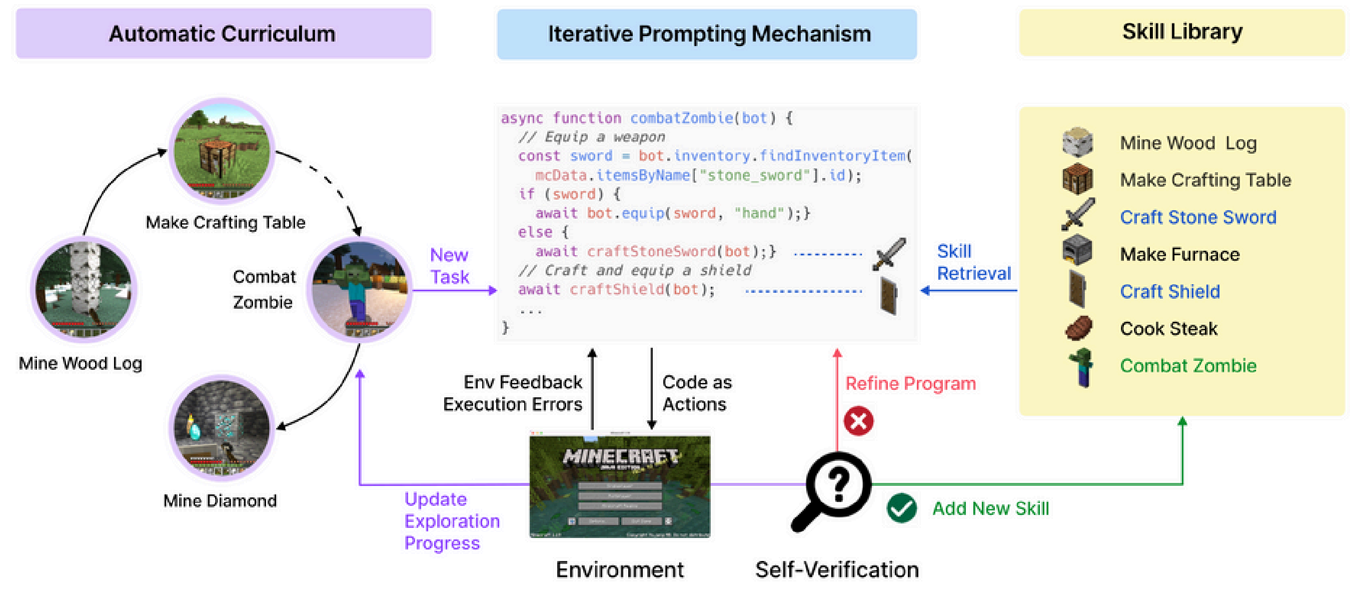


Figure 2: VOYAGER consists of three key components: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

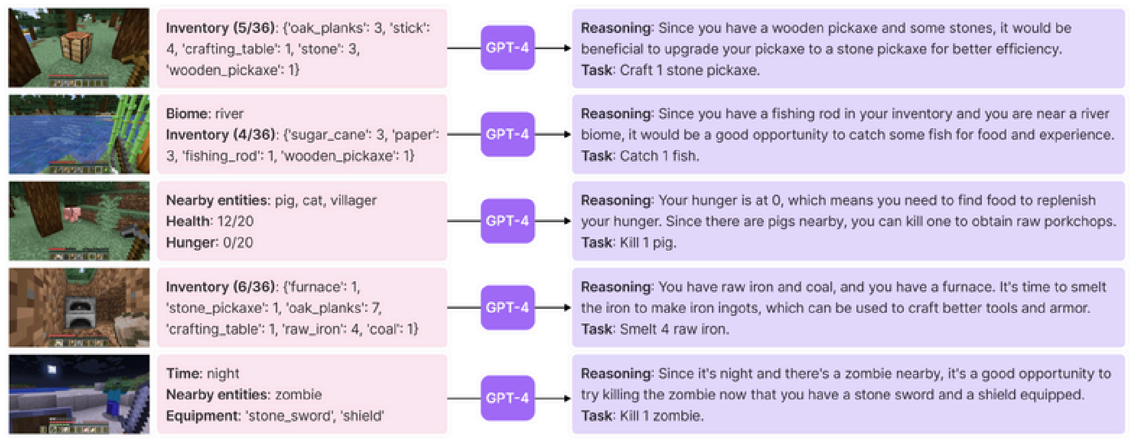


Figure 3: Tasks proposed by the automatic curriculum. We only display the partial prompt for brevity. See Appendix, Sec. A.3 for the full prompt structure.

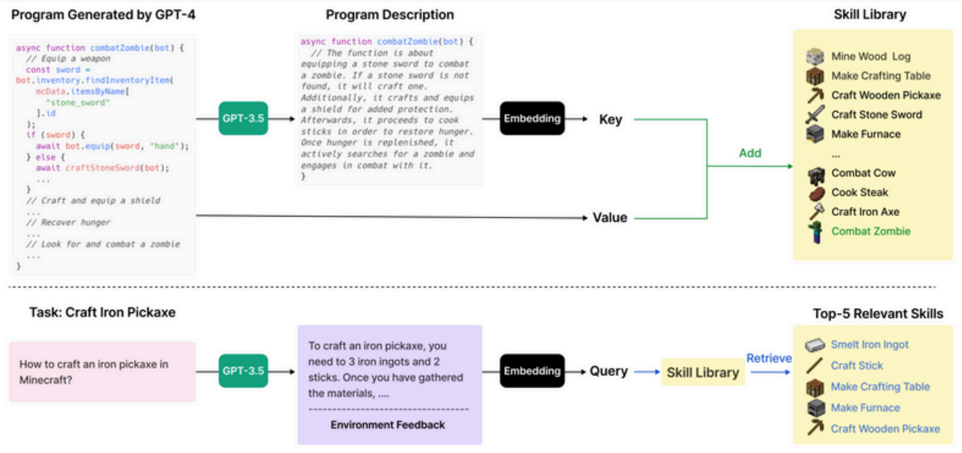


Figure 4: Skill library. **Top: Adding a new skill.** Each time GPT-4 generates and verifies a new skill, we add it to the skill library, represented by a vector database. The key is the embedding vector of the program description (generated by GPT-3.5), while the value is the program itself. **Bottom: Skill retrieval.** When faced with a new task proposed by the automatic curriculum, we first leverage GPT-3.5 to generate a general suggestion for solving the task, which is combined with environment feedback as the query context. Subsequently, we perform querying to identify the top-5 relevant skills.

WANG, Guanzhi, XIE, Yuqi, JIANG, Yunfan, et al. Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291, 2023. [link](), [code]()

SECANT: Self-Expert Cloning for Zero-Shot Generalization of Visual Policies [Fan+ (NVIDIA), PLMR21]

- **Objective**: The main goal is to achieve zero-shot generalization to unseen visual environments. The approach aims to improve the robustness of policy learning against environmental variations in visual appearance while maintaining high performance.
- **Methodology**: SECANT employs a two-stage training process:
  - **Expert Policy Training**: An expert policy is trained using RL with weak augmentations like random cropping. This step focuses on achieving high-performance policies.
  - **Student Policy Distillation**: A student network learns to mimic the expert policy but with strong augmentations (e.g., cutout-color, Gaussian noise, Mixup, Cutmix). This helps the student develop robust visual representations, making it less sensitive to environmental variations compared to the expert.
- **Zero-Shot Generalization:** SECANT demonstrates significant improvements in zero-shot generalization across various domains, including DeepMind Control, robotic manipulation, vision-based autonomous driving, and indoor object navigation. The method outperforms state-of-the-art approaches, often by wide margins, showcasing its effectiveness in handling unseen visual environments without the need for further training or adaptation at test time.

SECANT: Self-Expert Cloning for Zero-Shot Generalization of Visual Policies

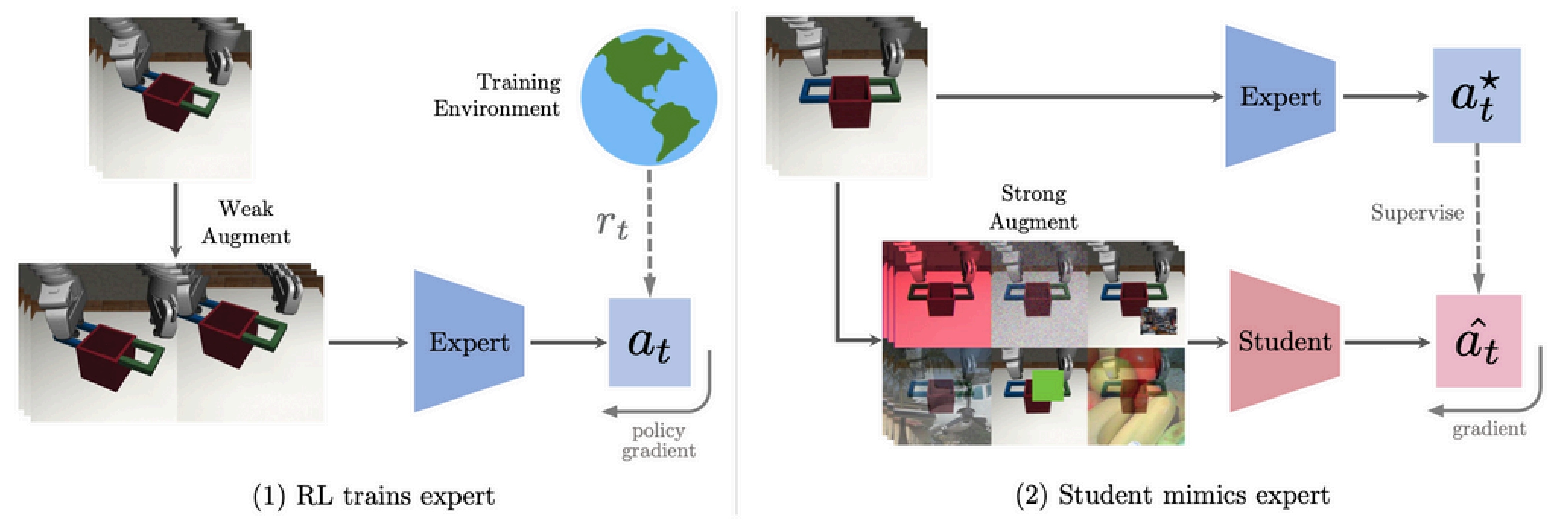


Figure 2. Algorithm overview. SECANT training is split into two stages. **Left, stage 1**: expert policy is trained by RL with weak augmentation (random cropping). **Right, stage 2**: student receives ground-truth action supervision from the expert at every time step, conditioned on the same observation but with strong augmentations, such as cutout-color, Gaussian noise, Mixup, and Cutmix. The student learns robust visual representations invariant to environment distractions, while maintaining high policy performance.

Algorithm 1 SECANT: Self-Expert Cloning

π_e, π_s: randomly initialized expert and student policies
$\mathcal{F}_{weak}, \mathcal{F}_{strong}$: sets of image augmentations
$\mathcal{B}$: experience replay buffer
for t in $1, \ldots, T_{RL}$ **do**
  Sample experience batch $\tau_t = (o_t, a_t, o_{t+1}, r) \sim \mathcal{B}$
  Sample weak augmentation $f \sim \mathcal{F}_{weak}$
  Augment $o_t = f(o_t); o_{t+1} = f(o_{t+1})$
  Update π_e to minimize $\mathcal{L}_{RL}(\tau_t)$
end for
Roll out π_e to collect an initial dataset $\mathcal{D}$ of trajectories
for t in $1, \ldots, T_{imitate}$ **do**
  Sample observation batch $o \sim \mathcal{D}$
  Sample strong augmentation $f \sim \mathcal{F}_{strong}$
  Update π_s to minimize $\|\pi_s(f(o)) - \pi_e(o)\|_F$
  Roll out π_s for one environment step and add to the dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{o_s\}$
end for

FAN, Linxi, WANG, Guanzhi, HUANG, De-An, et al. Secant: Self-expert cloning for zero-shot generalization of visual policies. arXiv preprint arXiv:2106.09678, 2021. **link**, **code**

Open X-Embodiment: Robotic Learning Datasets and RT-X Models [Padalkar+ (Google DeepMind), arxiv preprint23]

- **Objective**: The study aims to assess whether large-scale, diverse datasets can enable generalist robotic policies that adapt efficiently to various robots, tasks, and environments.
- **Methodology**: It introduces the RT-X models, trained on data from multiple robotic platforms, and an Open X-Embodiment Dataset, assembled from 22 different robots demonstrating 527 skills across 160,266 tasks.
- **Significance in Robotic Learning**: The paper demonstrates that high-capacity models trained on this dataset, termed RT-X, show positive transfer and enhanced capabilities across multiple robotic platforms by leveraging experiences from different robots. This approach marks a significant advancement in the field of robotics, pushing the boundaries of robotic learning towards more versatile and adaptable systems.

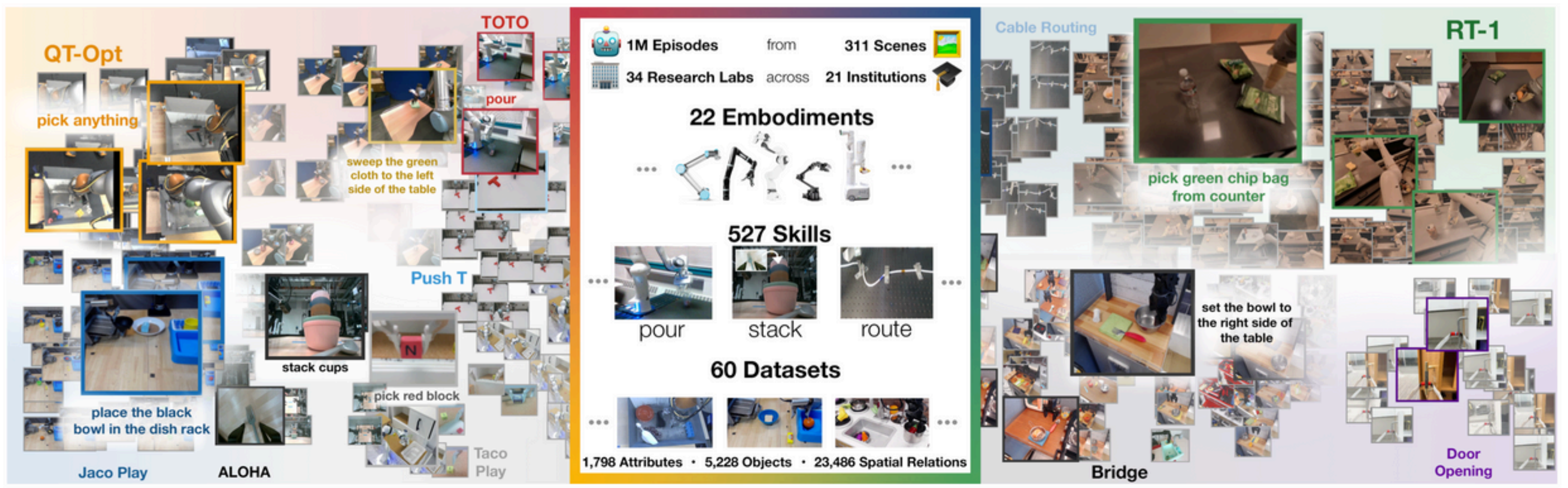


Fig. 1: We propose an open, large-scale dataset for robot learning curated from 21 institutions across the globe. The dataset represents diverse behaviors, robot embodiments and environments, and enables learning generalized robotic policies.

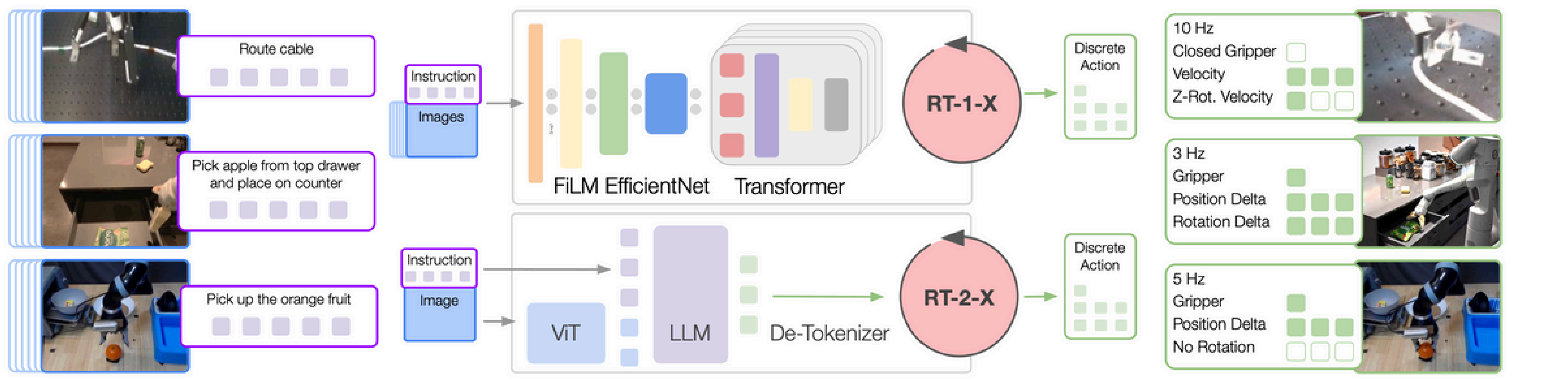


Fig. 3: RT-1-X and RT-2-X both take images and a text instruction as input and output discretized end-effector actions. RT-1-X is an architecture designed for robotics, with a FiLM [116] conditioned EfficientNet [117] and a Transformer [118]. RT-2-X builds on a VLM backbone by representing actions as another language, and training action text tokens together with vision-language data.

PADALKAR, Abhishek, POOLEY, Acorn, JAIN, Ajinkya, et al.Open X-Embodiment: Robotic learning datasets and RT-X models. arXiv preprint arXiv:2310.08864, 2023. [link](), [code]()

PaLM-E: An Embodied Multimodal Language Model [Driess+ (Robotics at Google), arxiv preprint23]

- **Objective**: Develop an embodied multimodal language model, PaLM-E, that integrates continuous sensor data from embodied agents, enabling it to perform various tasks, from robotic manipulation planning to visual question answering.
- **Methodology**: PaLM-E uses multimodal sentences combining text and continuous inputs like images or 3D representations. It's trained end-to-end on tasks including robotic manipulation, visual QA, and language tasks, leveraging a large-scale dataset encompassing diverse domains.
- **Innovative Approach in Embodied AI**: The paper demonstrates how PaLM-E can address a variety of embodied reasoning tasks across multiple embodiments, showing positive transfer from diverse joint training. PaLM-E's integration of vision and language with embodied reasoning represents a significant advancement in AI, allowing for more efficient and adaptable agents in real-world applications.

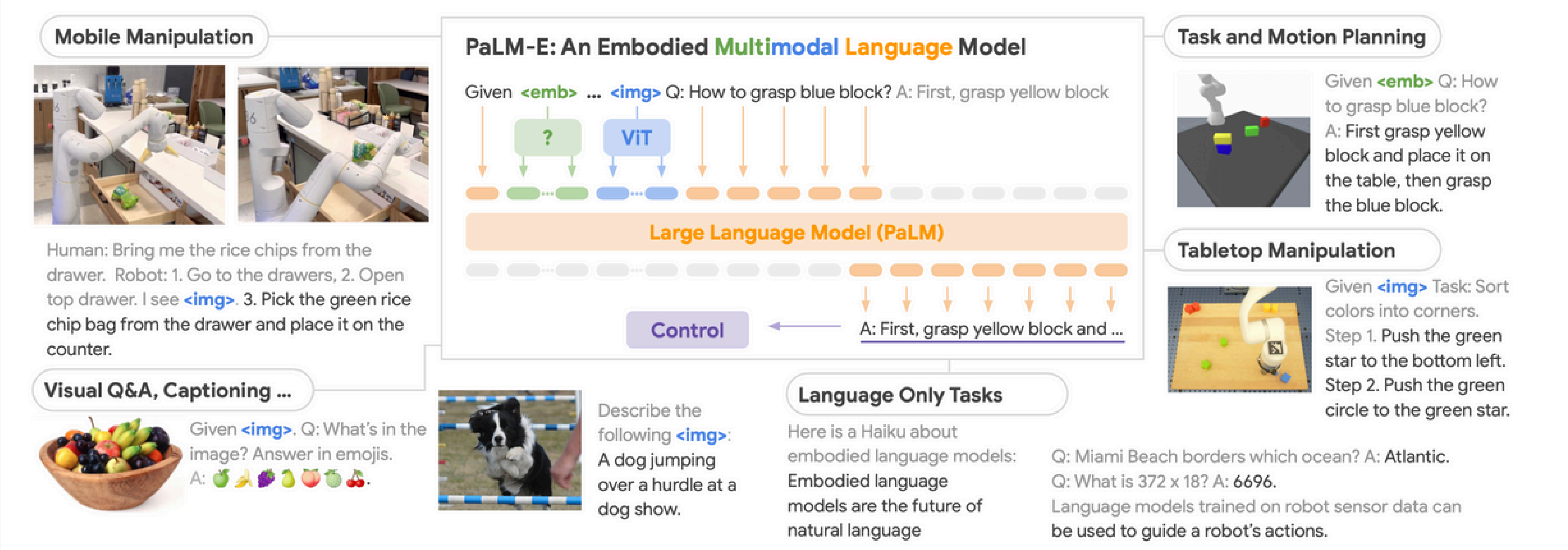
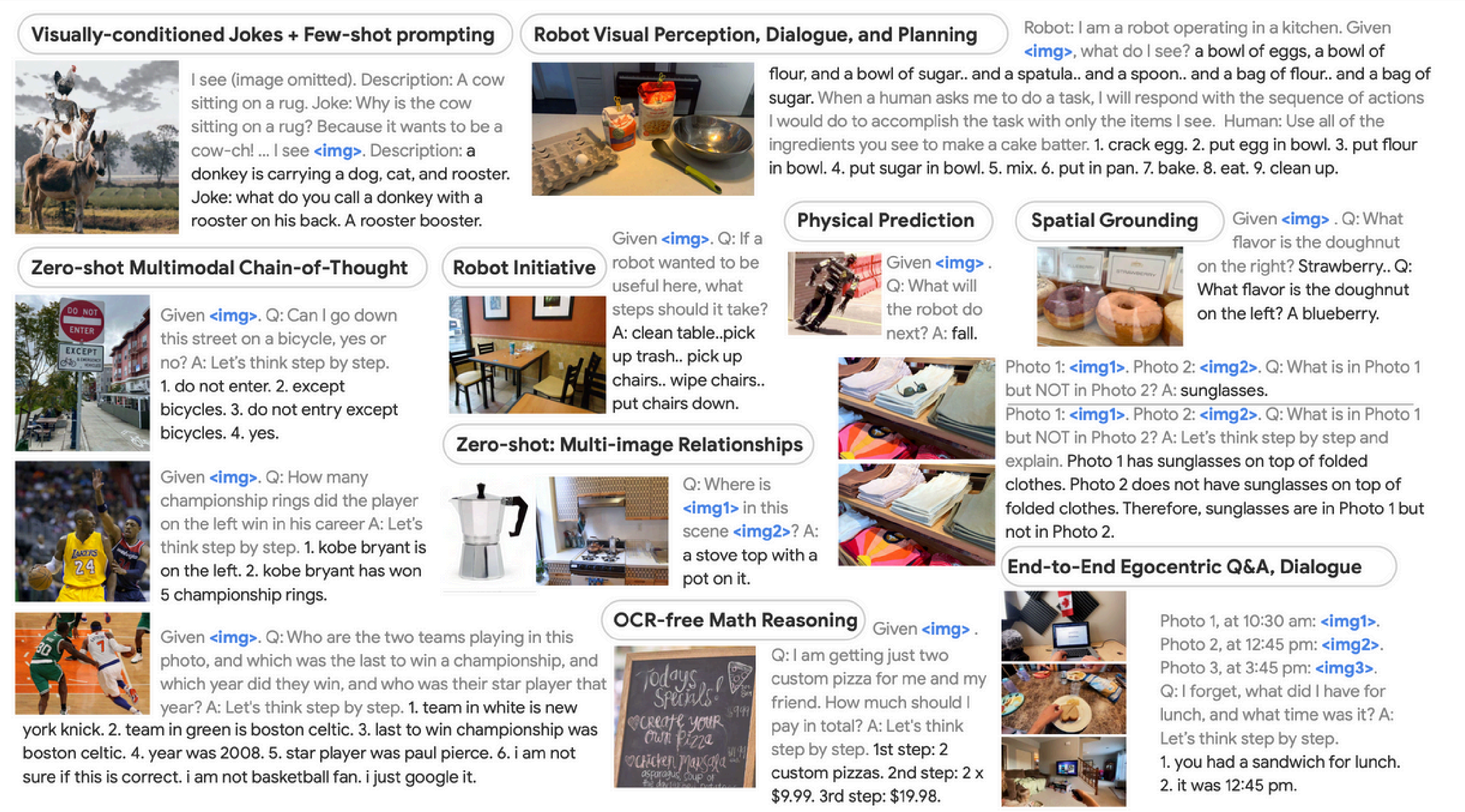


Figure 1: PaLM-E is a single general-purpose multimodal language model for embodied reasoning tasks, visual-language tasks, and language tasks. PaLM-E *transfers* knowledge from visual-language domains into embodied reasoning – from robot planning in environments with complex dynamics and physical constraints, to answering questions about the observable world. PaLM-E operates on *multimodal sentences*, i.e. sequences of tokens where inputs from arbitrary modalities (e.g. images, neural 3D representations, or states, in green and blue) are inserted alongside text tokens (in orange) as input to an LLM, trained end-to-end.



DRIESS, Danny, XIA, Fei, SAJJADI, Mehdi SM, et al. Palm-e: An embodied multimodal language model. arXiv preprint arXiv:2303.03378, 2023. link

BridgeData V2: A Dataset for Robot Learning at Scale [(Walke+ (UC Berkeley), CoRL23]

- **Objective**: The aim is to facilitate robot learning research by providing a dataset supporting broad generalization to novel tasks, environments, and institutions. It emphasizes task conditioning through goal images or natural language instructions.
- **Methodology**: BridgeData V2 comprises 60,096 trajectories across 24 environments with 13 skills. It's designed for multi-task learning and supports various robot learning methods.
- **Dataset Significance**: The dataset's diversity enables better generalization and its compatibility with different learning methods demonstrates its utility for scalable robot learning.

Figure 1 (BridgeData V2) We propose a large-scale robotic manipulation dataset containing 60,096 trajectories across 24 environments. The dataset includes skills such as pick-and-place, pushing, sweeping, stacking, folding, and more. Portions of the data include multiple camera views and depth data, and all of the data includes natural language labels.

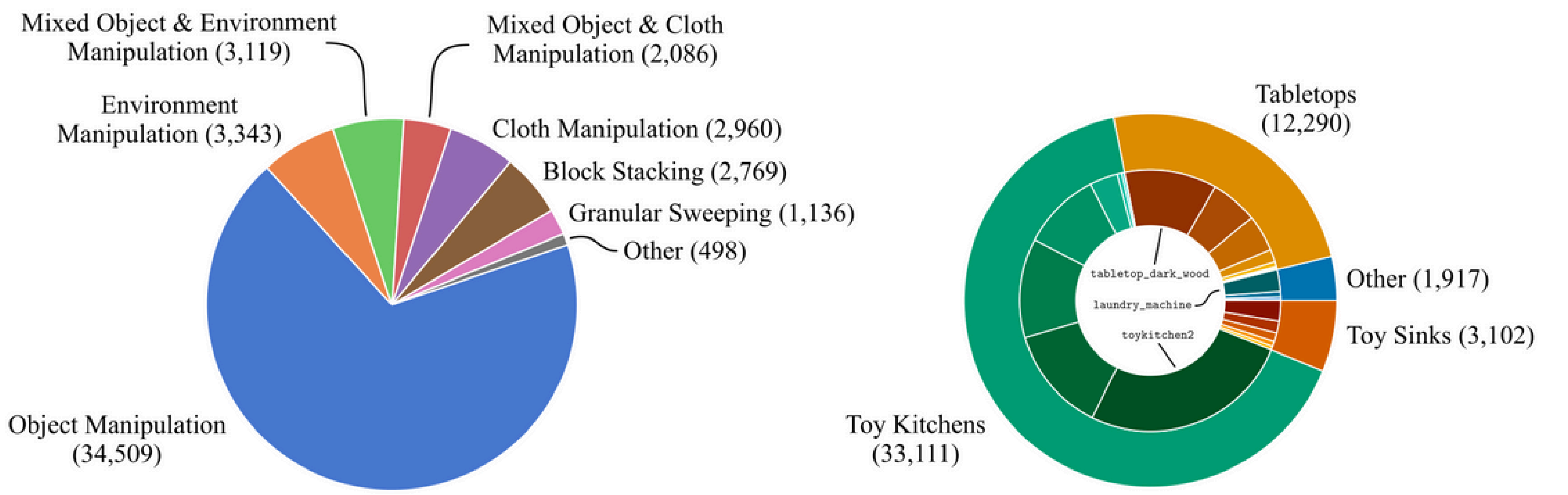


Figure 3 (Dataset composition) (L) The various types of tasks in BridgeData V2. The majority of the data comes from foundational object manipulation tasks, such as pick-and-place, pushing, and sweeping. Additional data comes from environment manipulation, which includes opening and closing doors and drawers. The remaining data comes from more complex tasks, such as stacking blocks, folding cloths, and sweeping granular media. Some segments of the data contain mixtures of these categories. **(R)** The 24 environments in BridgeData V2, grouped into 4 categories. The majority of the data comes from 7 distinct toy kitchens, which include some combination of sinks, stoves, and microwaves. The remaining environments come from diverse sources, including various tabletops, standalone toy sinks, a toy laundry machine, and more.

WALKE, Homer Rich, BLACK, Kevin, ZHAO, Tony Z., et al.Bridgedata v2: A dataset for robot learning at scale. In : Conference on Robot Learning. PMLR, 2023. p. 1723-1736. [link], [code]

R3M: A Universal Visual Representation for Robot Manipulation [Nair+ (Stanford Univ.), CoRL22]

- **Objective**: Explore the potential of pre-trained visual representations from human videos in enabling data-efficient learning for robotic manipulation.
- **Methodology**: Focuses on pre-training a visual representation (R3M) using the Ego4D human video dataset with techniques like time-contrastive learning, video-language alignment, and L1 sparsity penalty, aiming for a compact and task-relevant representation.
- **Innovation in Robotic Manipulation**: Demonstrates that R3M improves task success significantly compared to training from scratch or using other state-of-the-art visual representations. R3M enables a robot to learn a range of manipulation tasks in real-world settings with minimal demonstrations, suggesting its potential as a standard vision model for robotic manipulation.

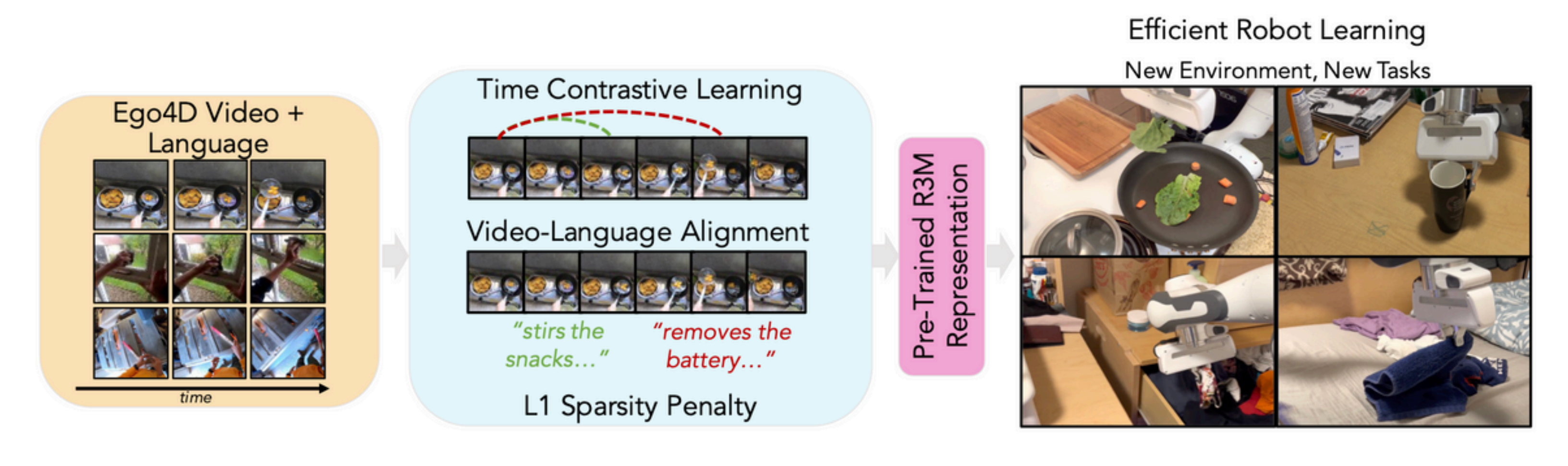


Figure 1: **Pre-Training Reusable Representations for Robot Manipulation (R3M)**: We pre-train a visual representation using diverse human video datasets like Ego4D [16], and study its effectiveness for downstream robot manipulation tasks. Our representation model, R3M, is trained using a combination of time-contrastive learning, video-language alignment, and an L1 sparsity penalty. We find that R3M enables data efficient imitation learning across several simulated and real-world robot manipulation tasks.

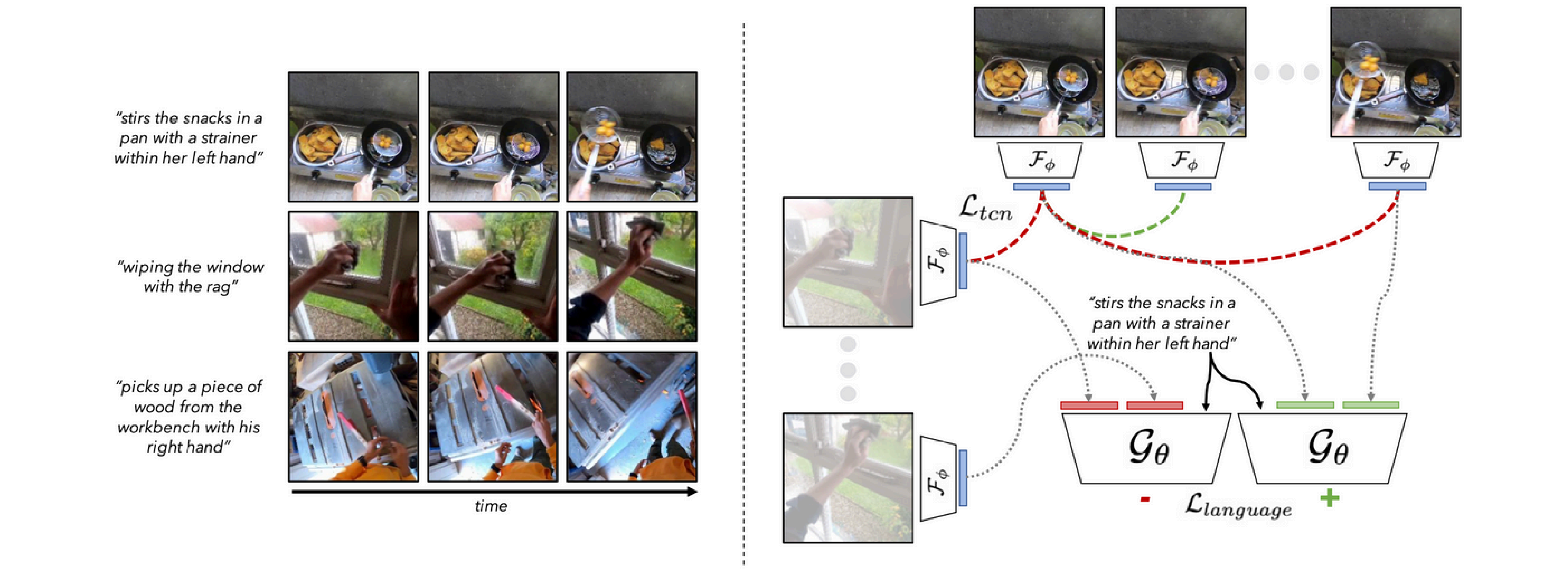


Figure 2: **Ego4D [16] Video and Language (left)**. Sample frames and associated language from Grauman et al. [16] used for training R3M. **R3M Training (right)**. We train R3M with time contrastive learning, encouraging states closer in time to be closer in embedding space and video-language alignment to encourage the embeddings to capture semantically relevant features.

VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models [Huang+ (Stanford Univ.), CoRL23]

- **Objective**: Develop VoxPoser, a framework that utilizes large language models (LLMs) to guide robotic manipulation tasks via language instructions. The goal is to enhance robot interaction and task execution using natural language.
- **Methodology**: VoxPoser combines LLMs with 3D value map composition for task planning and execution. It employs LLMs to interpret language instructions and generate actionable tasks. These are then translated into 3D value maps, guiding robots in physical environments.
- **Innovation in Robotic Manipulation**: The paper showcases VoxPoser's ability to understand and execute a wide range of manipulation tasks through language instructions. It demonstrates how LLMs can be utilized to extract actionable knowledge for robotic manipulation without extensive task-specific training, offering flexibility and efficiency in robotic applications.

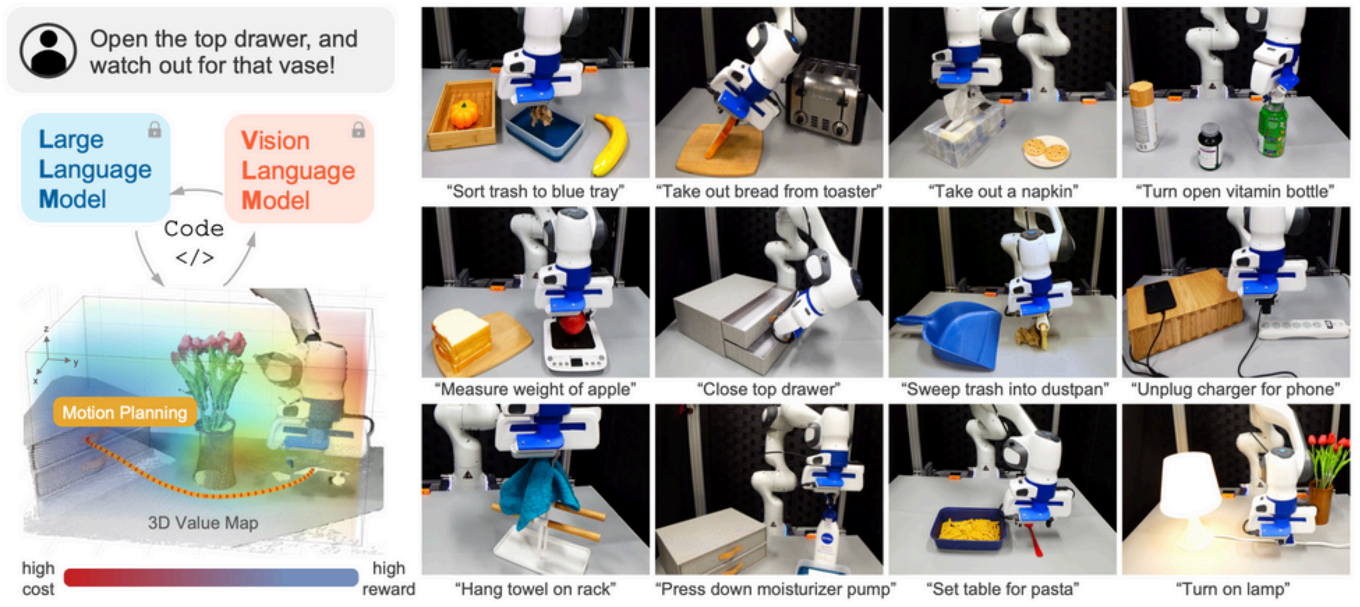


Figure 1: VoxPoser extracts language-conditioned **affordances** and **constraints** from LLMs and grounds them to the perceptual space using VLMs, using a code interface and without additional training to either component. The composed map is referred to as a 3D value map, which enables **zero-shot** synthesis of trajectories for large varieties of everyday manipulation tasks with an **open-set of instructions** and an **open-set of objects**.

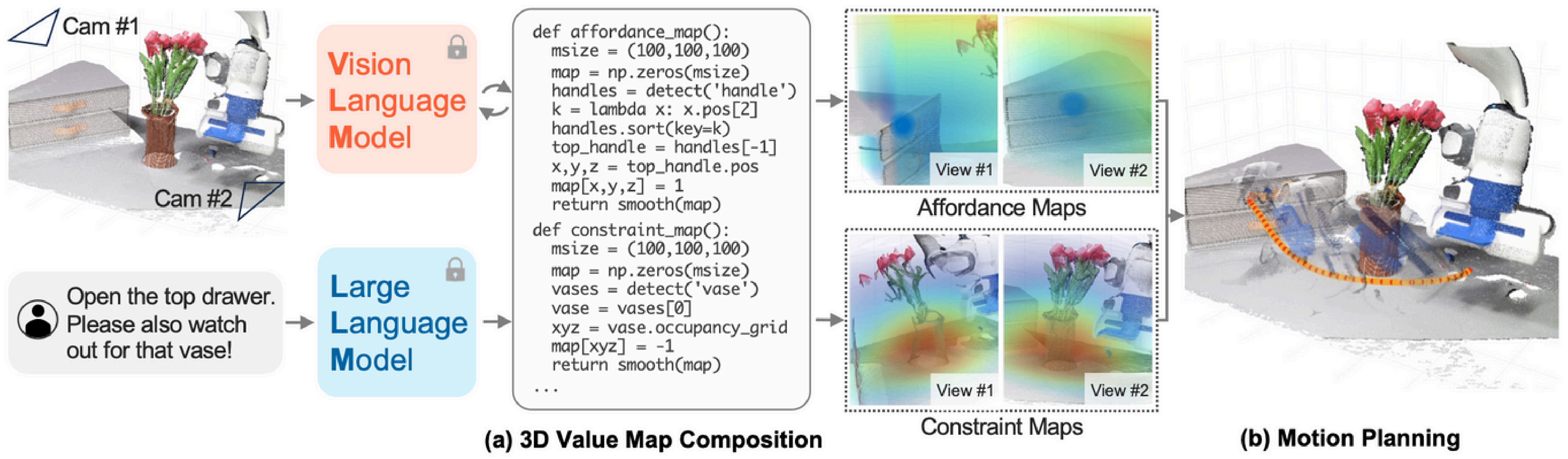


Figure 2: Overview of VoxPoser. Given the RGB-D observation of the environment and a language instruction, LLMs generate code, which interacts with VLMs, to produce a sequence of 3D affordance maps and constraint maps (collectively referred to as value maps) grounded in the observation space of the robot **(a)**. The composed value maps then serve as objective functions for motion planners to synthesize trajectories for robot manipulation **(b)**. The entire process does not involve any additional training.

HUANG, Wenlong, WANG, Chen, ZHANG, Ruohan, et al.Voxposer: Composable 3d value maps for robotic manipulation with language models. arXiv preprint arXiv:2307.05973, 2023. link, code

VIMA: General Robot Manipulation with Multimodal Prompts [Jiang+ (Stanford Univ.), ICML23]

- **Objective**: Develop VIMA, an agent for robotic manipulation, capable of interpreting multimodal prompts for a wide range of tasks, thus enabling more intuitive and flexible task specification in robotics.
- **Methodology**: VIMA utilizes a transformer-based architecture, processing multimodal prompts (a combination of textual and visual tokens) to generate motor actions. The system is tested on VIMA-BENCH, a new benchmark with diverse, procedurally-generated tasks for systematic evaluation.
- **Innovation in Robotic Task Specification**: VIMA's key contribution is its ability to unify various robot manipulation tasks into a single framework using multimodal prompts. This innovation simplifies task specification and allows for more generalizable, efficient, and scalable learning and execution in robotic systems.

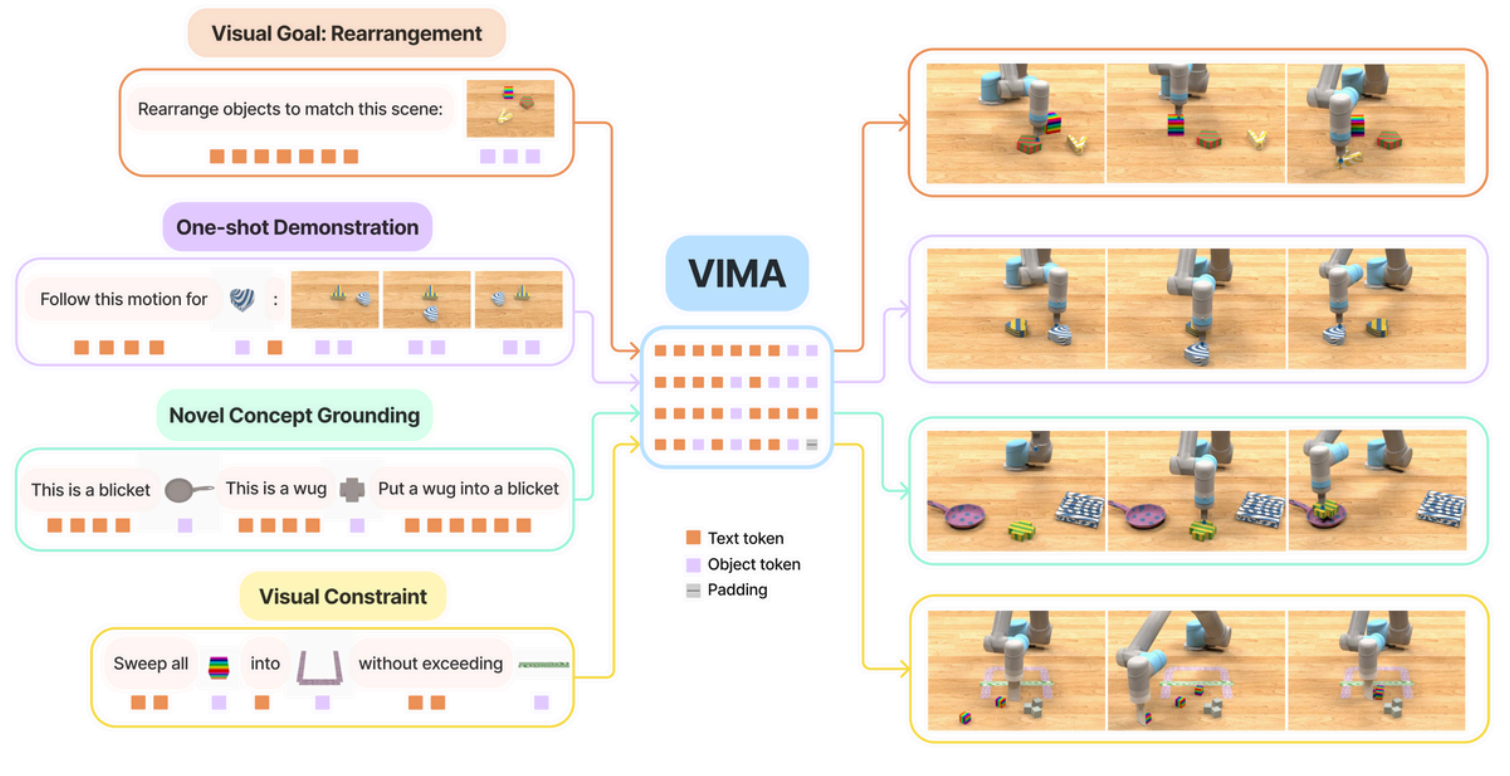


Figure 1: **Multimodal prompts for task specification.** We observe that many robot manipulation tasks can be expressed as *multimodal prompts* that interleave language and image/video frames. We introduce VIMA, an embodied agent capable of processing mulitimodal prompts (left) and controlling a robot arm to solve the task (right).

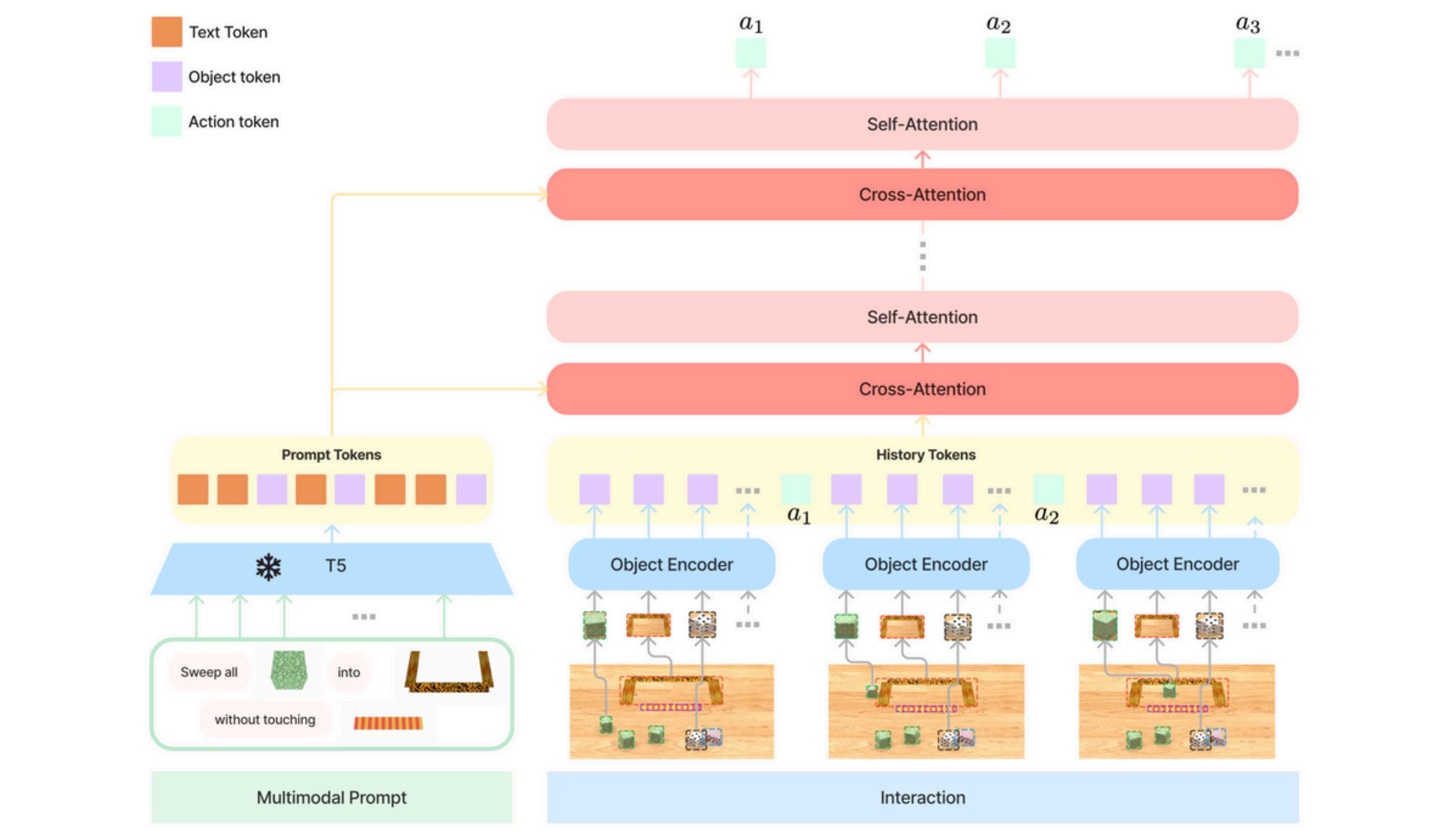


Figure 3: **VIMA Architecture.** We encode the multimodal prompts with a pre-trained T5 model, and condition the robot controller on the prompt through cross-attention layers. The controller is a causal transformer decoder consisting of alternating self and cross attention layers that predicts motor commands conditioned on prompts and interaction history.

JIANG, Yunfan, GUPTA, Agrim, ZHANG, Zichen, et al. Vima: General robot manipulation with multimodal prompts. arXiv, 2022.
link, code

Language to Rewards for Robotic Skill Synthesis [Yu+ (Google DeepMind), arxiv preprint23]

- **Objective**: The study aims to bridge the gap between high-level language instructions and low-level robotic actions by leveraging LLMs to specify reward functions that guide robotic behavior.
- **Methodology**: The approach involves a system with a Reward Translator, which interprets user input and transforms it into a reward specification. This is combined with a real-time optimizer, MuJoCo MPC, for interactive behavior creation, allowing immediate feedback and adjustment.
- **Innovation in Robotic Control**: The paper's innovation lies in using LLMs for reward function generation, enabling a more flexible and intuitive method for controlling robots. It demonstrates this method's effectiveness with various tasks on simulated quadruped and dexterous manipulator robots and validates it on real robot hardware. The system shows a significant improvement over traditional methods, handling complex manipulation skills and diverse locomotion tasks based on language instructions.

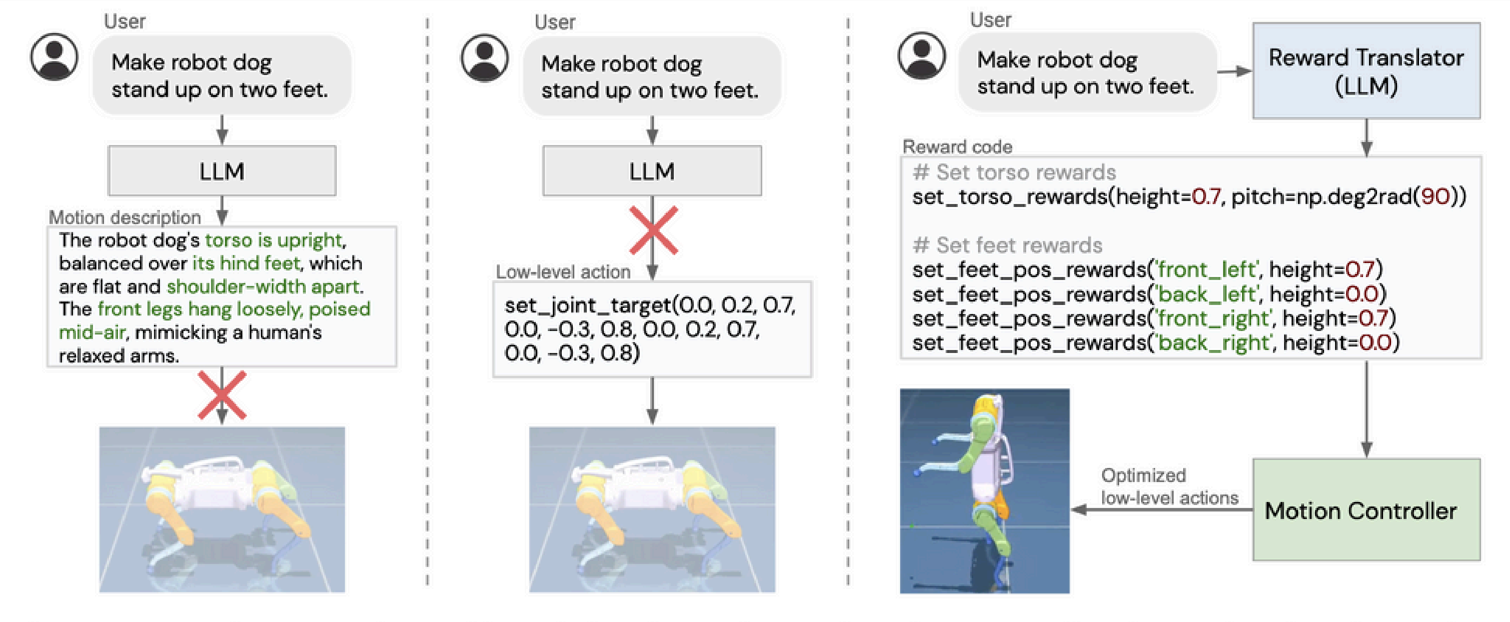


Figure 1: LLMs have some internal knowledge about robot motions, but cannot directly translate them into actions (left). Low-level action code can be executed on robots, but LLMs know little about them (mid). We attempt to bridge this gap, by proposing a system (right) consisting of the Reward Translatorthat interprets the user input and transform it into a reward specification. The reward specification is then consumed by a Motion Controller that interactively synthesizes a robot motion which optimizes the given reward.

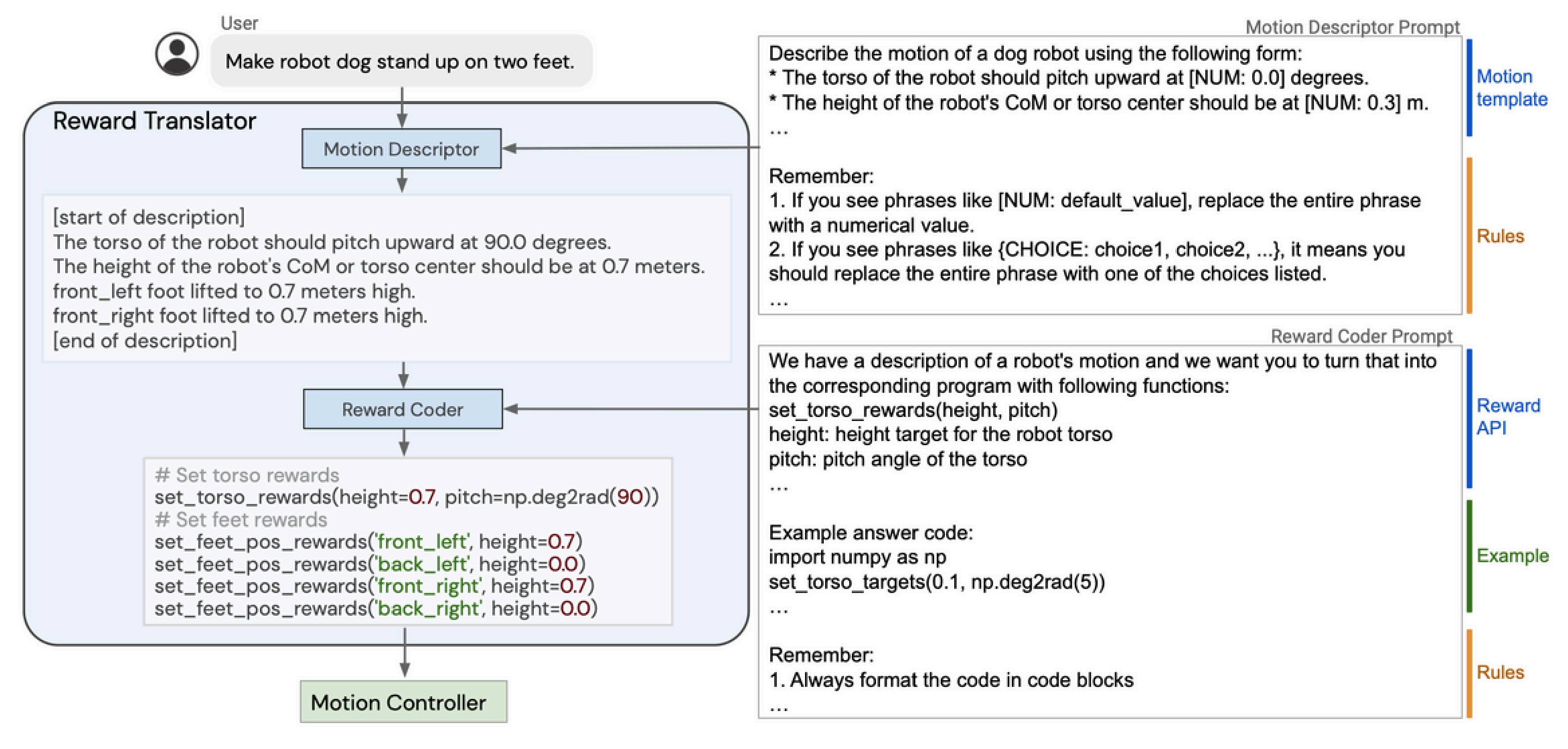


Figure 2: Detailed dataflow of the Reward Translator. A *Motion Descriptor* LLM takes the user input and describe the user-specified motion in natural language, and a *Reward Coder* translates the motion into the reward parameters.

Code as Policies: Language Model Programs for Embodied Control [Liang+ (Robotics at Google), IEEE23]

- **Objective**: Develop a method where robots translate natural language commands into policy code, using large language models (LLMs) trained on code-completion.
- **Methodology**: Utilizes code-writing LLMs to process natural language commands and autonomously generate robot policy code. This involves integrating classic logic structures and third-party libraries for spatial-geometric reasoning and precise control.
- **Innovative Robotic Control Approach:** The paper presents a unique way of leveraging LLMs for robotic manipulation and control. This approach allows robots to interpret natural language instructions and generate policies that react to sensory inputs, showcasing the potential of LLMs in complex, real-world robotic applications.

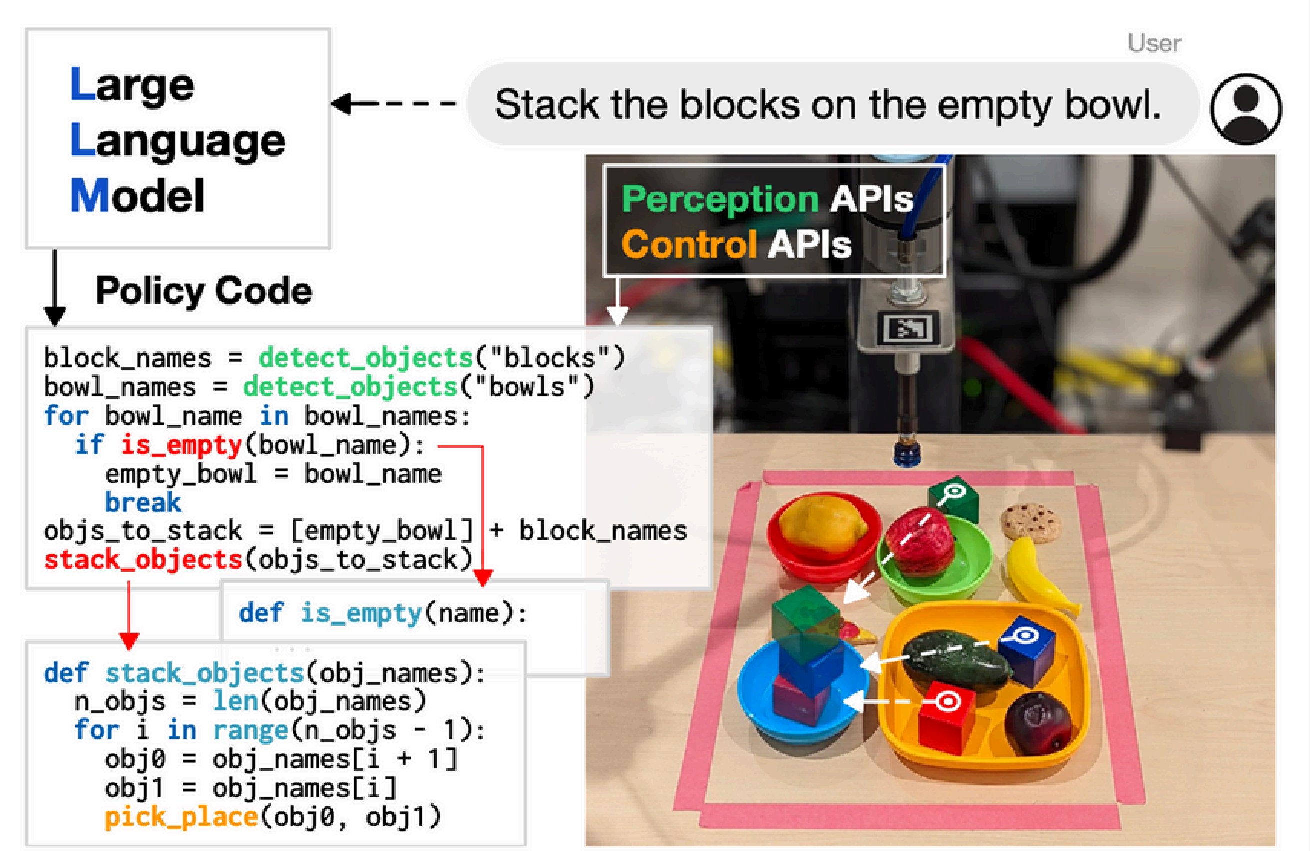


Fig. 1: Given examples (via few-shot prompting), robots can use code-writing large language models (LLMs) to translate natural language commands into robot policy code which process perception outputs, parameterize control primitives, recursively generate code for undefined functions, and generalize to new tasks.

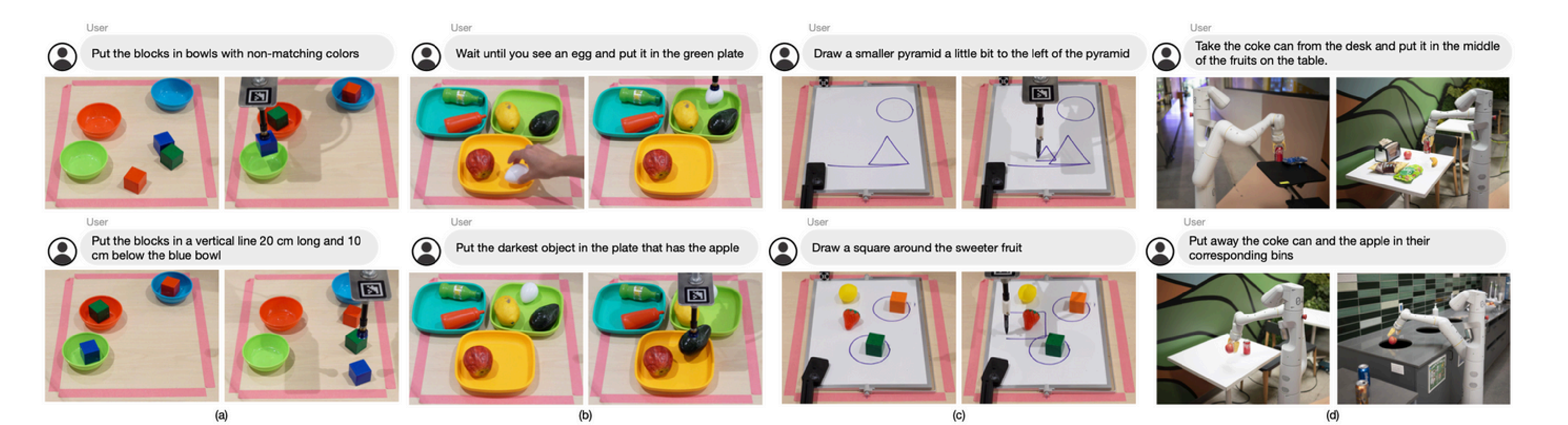


Fig. 2: Code as Policies can follow natural language instructions across diverse domains and robots: table-top manipulation (a)-(b), 2D shape drawing (c), and mobile manipulation in a kitchen with robots from Everyday Robots (d). Our approach enables robots to perform spatial-geometric reasoning, parse object relationships, and form multi-step behaviors using off-the-shelf models and few-shot prompting with no additional training. See full videos and more tasks at code-as-policies.github.io

LIANG, Jacky, HUANG, Wenlong, XIA, Fei, et al. Code as policies: Language model programs for embodied control. In : 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023. p. 9493-9500., [link], [code]

Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning [Makoviychuk+ (NVIDIA), arxiv preprint21]

- **Objective**: Develop a GPU-accelerated training platform for robotics tasks, enhancing simulation and training efficiency in robotics RL.
- **Methodology**: Isaac Gym uses a GPU-based physics engine and a PyTorch tensor API, allowing direct data transfer between physics simulation and neural network policy training without CPU bottlenecks.
- **Simulation and Training Performance:** The platform demonstrates significant speed improvements in training complex robotics tasks on a single GPU, surpassing traditional CPU-based simulators in efficiency and scalability. This advancement opens new possibilities for sophisticated and efficient robotic learning systems.

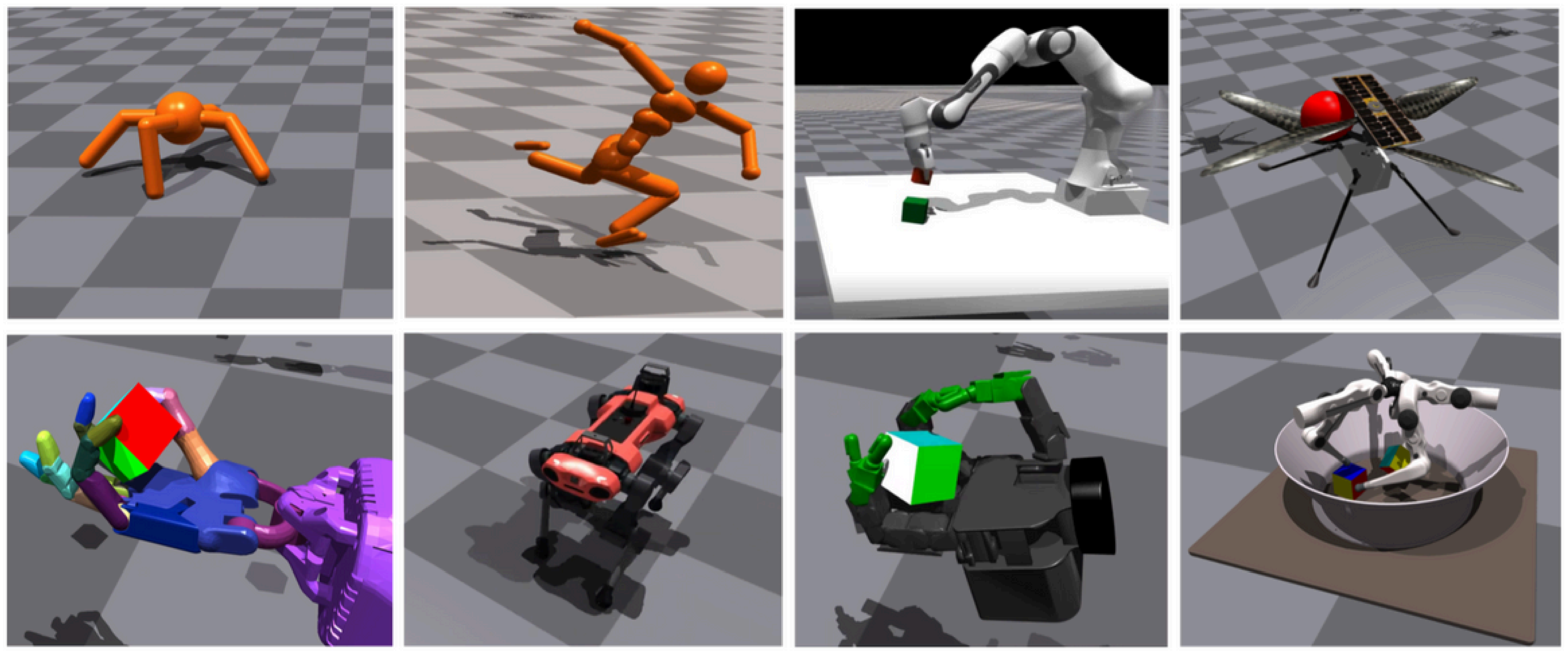


Figure 1: Isaac Gym allows high performance training on a variety of robotics environments. We benchmark on 8 different environments that offer a wide range of complexity and show the strengths of the simulator in blazing fast policy training on a single GPU. *Top*: Ant, Humanoid, Franka-cube-stack, Ingenuity. *Bottom*: Shadow Hand, ANYmal, Allegro, TriFinger.

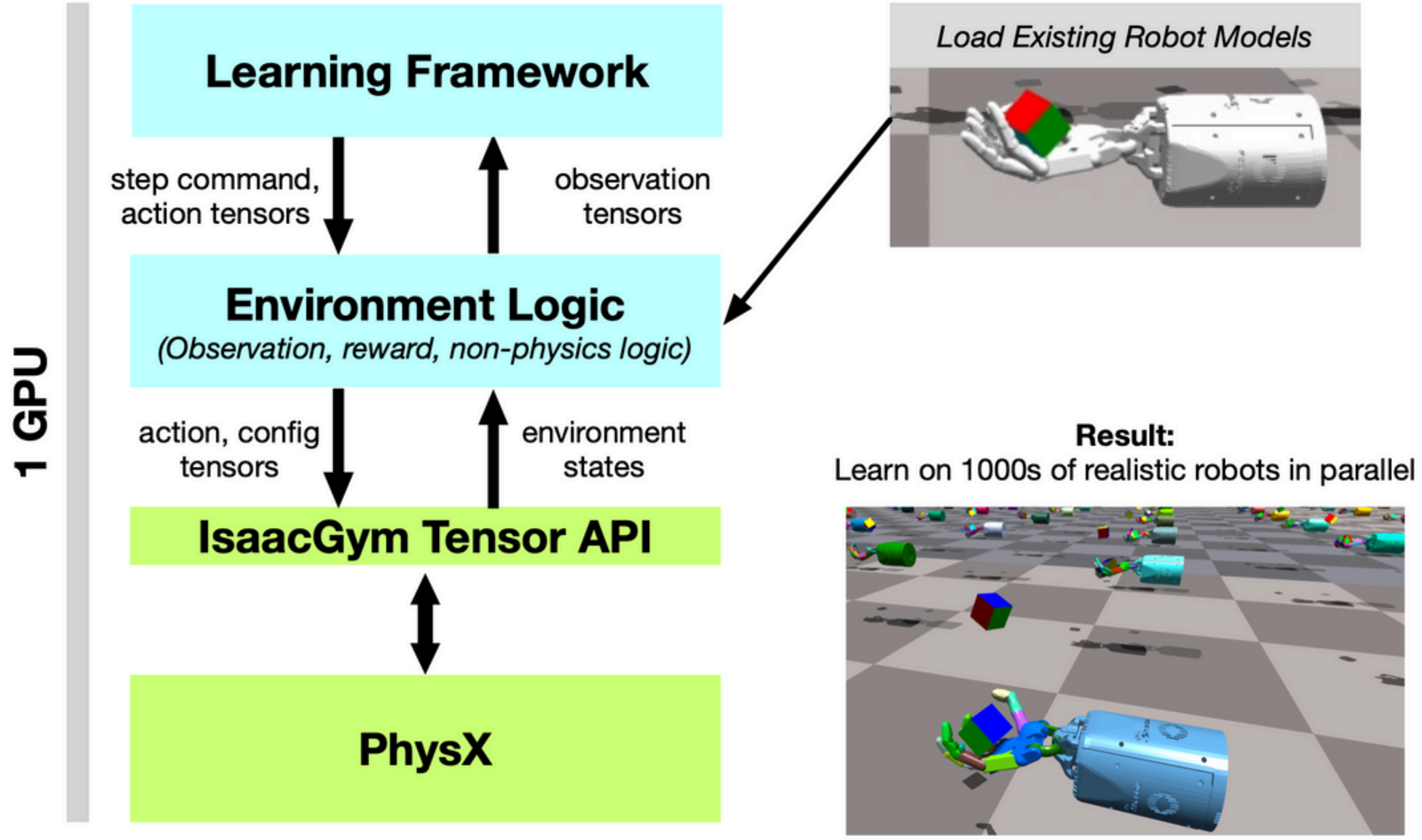


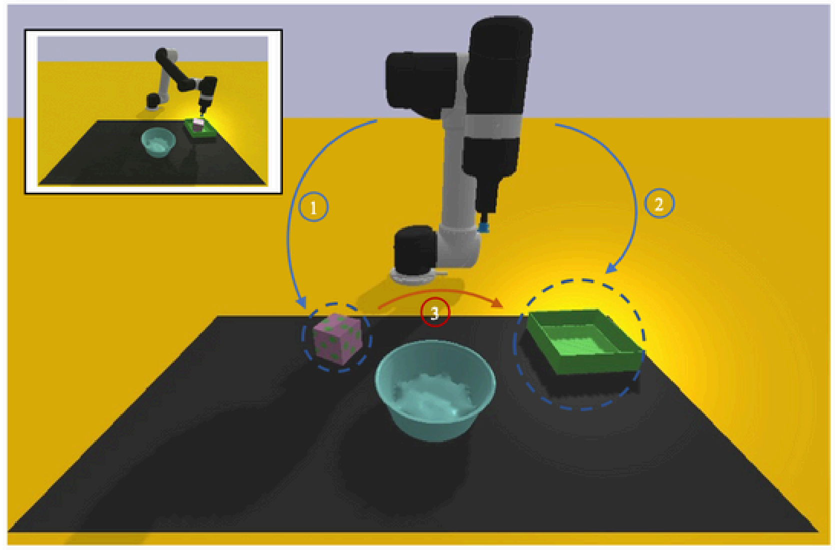
Figure 2: An illustration of the Isaac Gym pipeline. The Tensor API provides an interface to Python code to step the PhysX backend, as well as get and set simulator states, directly on the GPU, allowing a 100-1000x speedup in the overall RL training pipeline while providing high-fidelity simulation and the ability to interface with existing robot models.

MAKOVIYCHUK, Viktor, WAWRZYNIAK, Lukasz, GUO, Yunrong, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. arXiv preprint arXiv:2108.10470, 2021., **link**

Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model [Huang+ (Shanghai Univ., ICLR24]

- **Objective**: Develop a system (Instruct2Act) using LLMs to translate diverse instructions into executable robotic actions, enhancing flexibility in robotic manipulation tasks.
- **Methodology**: The approach integrates language and visual understanding, using LLM-generated Python programs to form a perception-action loop. This includes using APIs to access various foundation models (like SAM and CLIP) for object identification and classification.
- **Innovation in Robotic Task Execution:** Instruct2Act demonstrates adaptability in handling various instruction types and inputs, translating complex, high-level instructions into precise robotic actions. The framework is tested in different scenarios within tabletop manipulation domains, showing significant performance improvement over existing methods, particularly in zero-shot settings.

: Put the polka dot block into the green container.



(a) Robots are able to execute instructions that are provided as input in natural language.

File IO		GetObsImage( ) SaveFailureImage( )
Core Modules	Perception	SAM( ) CLIPRetrieval( ) GetObjMatch( )
	Action	Pixel2Loc( ) PickPlace( ) RearrangeActions( )
Robotic API		RobotExecution( ) SpeedSet( )

(b) Module examples utilized in Instruct2Act. The modules' definitions are hierarchical and aligned with the robotic system design.

Figure 1: A robotic task (a) is executed through the invocation of several modules (b) in Instruct2Act.

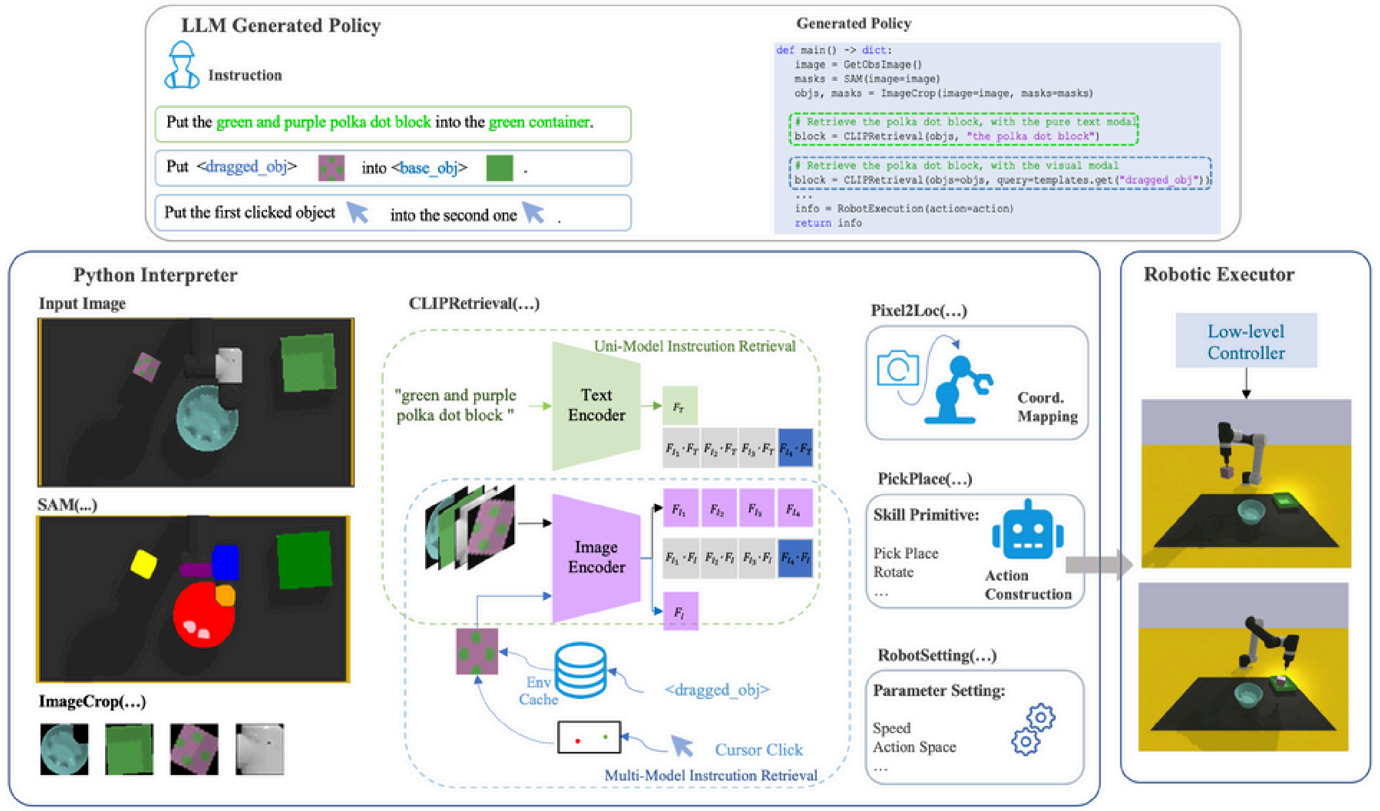


Figure 2: The paradigm of our proposed Instruct2Act framework. Starting with the task instruction, the framework utilizes LLM to generate executable code that invokes visual foundation models with APIs to recognize the environment. With recognized object semantic information, we generate plausible actions that are sent to the low-level controller to execute the task. The instructions in green and blue stand for pure-language and multimodal instructions respectively.

HUANG, Siyuan, JIANG, Zhengkai, DONG, Hao, et al.Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model. arXiv preprint arXiv:2305.11176, 2023., link

LLM+P: Empowering Large Language Models with Optimal Planning Proficiency [Liu+ (Texas Univ.), CoRR23]

- **Objective**: The paper's primary goal is to enable LLMs, such as GPT-4, to solve planning problems accurately by integrating them with classical planners.
- **Methodology**: LLM+P involves using LLMs to transform natural language descriptions of planning problems into the Planning Domain Definition Language (PDDL), then leveraging classical planners to find optimal solutions, which are translated back into natural language.
- **Innovative Approach in Planning:** This research demonstrates a significant advancement in the field of AI planning, where LLMs are used not as standalone solvers but as translators between natural language and formal planning languages. This allows for the generation of accurate and optimal plans, enhancing the planning capabilities of LLMs.

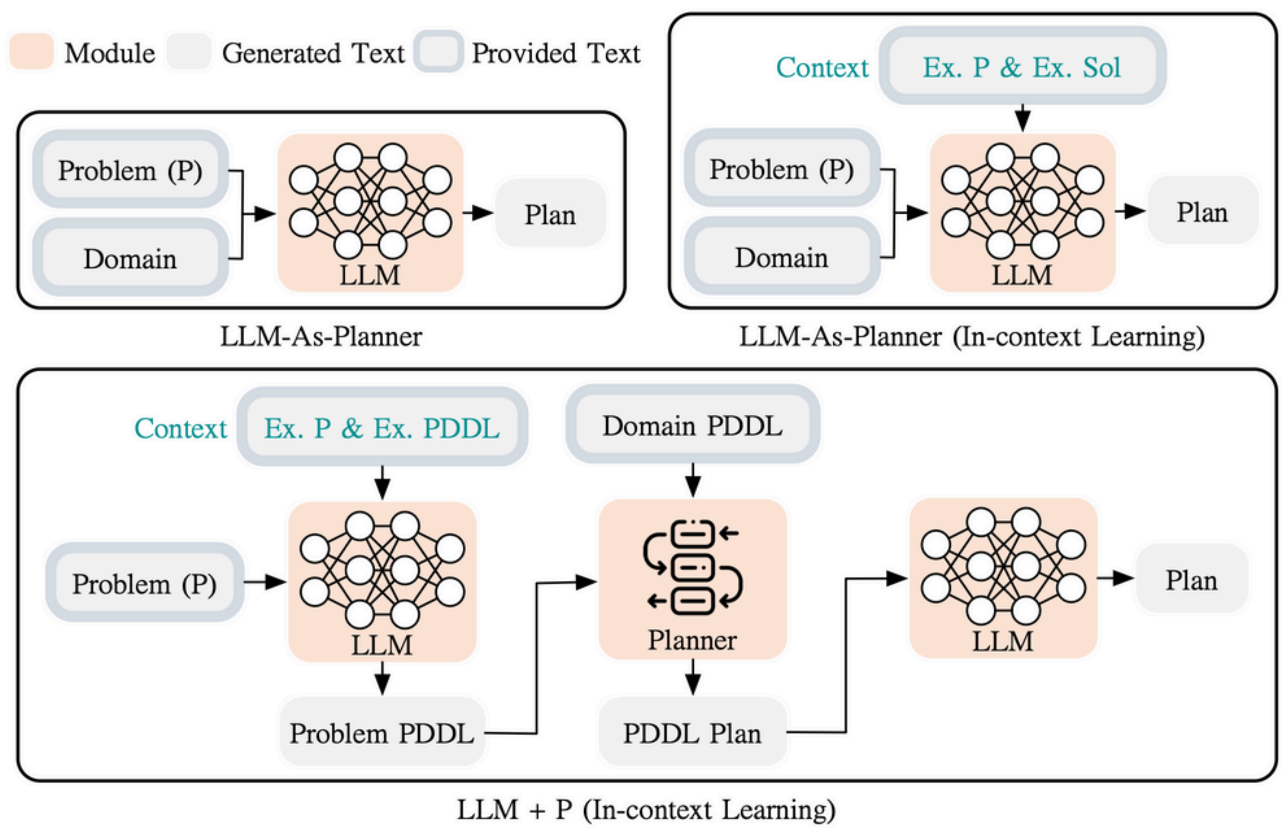


Fig. 1: LLM+P makes use of a large language model (LLM) to produce the PDDL description of the given problem, then leverages a classical planner for finding an *optimal* plan, then translates the raw plan back to natural language using the LLM again.

Tidy-Up Problem PDDL Generated by LLM+P

Problem (P): You are a home robot with one gripper. The distance between coffee table and side table is 10. The distance between coffee table and pantry is 20... You are at the coffee table. There is a mustard bottle... Your goal is to move objects to their destinations...

Problem PDDL generated by LLM+P:

```
(:objects coffee-table side-table
recycle-bin pantry - location
mustard-bottle soup-can - object)
(:init (= (total-cost) 0) (=
(distance coffee-table side-table)
10) (= (distance coffee-table
pantry) 20) ... (robot-at
coffee-table) (at mustard-bottle
coffee-table) (at soup-can
side-table) (hand-empty) )
(:goal (and (at mustard-bottle
pantry) (at soup-can recycle-bin)))
(:metric minimize (total-cost)) )
```

LIU, Bo, JIANG, Yuqian, ZHANG, Xiaohan, et al. Llm+ p: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477, 2023. link, code

Task and Motion Planning with Large Language Models for Object Rearrangement [Ding+ (Bighamton Univ.), IROS23]

- **Objective**: The research focuses on enhancing robot planning methods with common sense for object rearrangement, using LLMs to generate symbolic spatial relationships between objects, and grounding them in different geometric spatial relationships for task and motion planning.
- **Methodology**: LLM-GROP first uses an LLM to determine symbolic spatial relationships between objects (e.g., a fork and a knife being on the left and right, respectively). These relationships are then grounded into geometric spatial relationships, evaluated for feasibility by a motion planning system.
- **Innovation in Robotic Task Planning:** The study showcases LLM-GROP's ability to transform natural language commands into human-aligned object rearrangement in various environments. It demonstrates improvements in user satisfaction and maintains comparable cumulative action costs, showing LLM-GROP's practicality in real-world scenarios with a mobile manipulator.

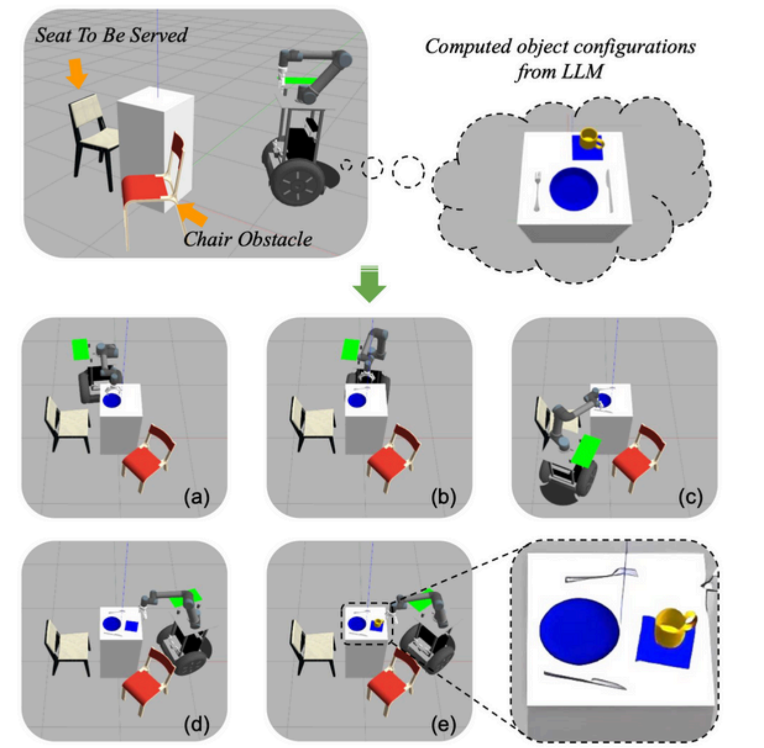


Fig. 1: A mobile manipulator is assigned the task of setting a table in a dining domain. The manipulator needs to arrange several tableware objects, including a knife, a fork, a plate, a cup mat, and a mug. These objects are available on the other tables, and there are also randomly generated obstacles (i.e., the red chair) that are not included in the pre-built map beforehand. The robot needs to compute feasible and efficient plans for rearranging the objects on the target table using both navigation and manipulation behaviors.

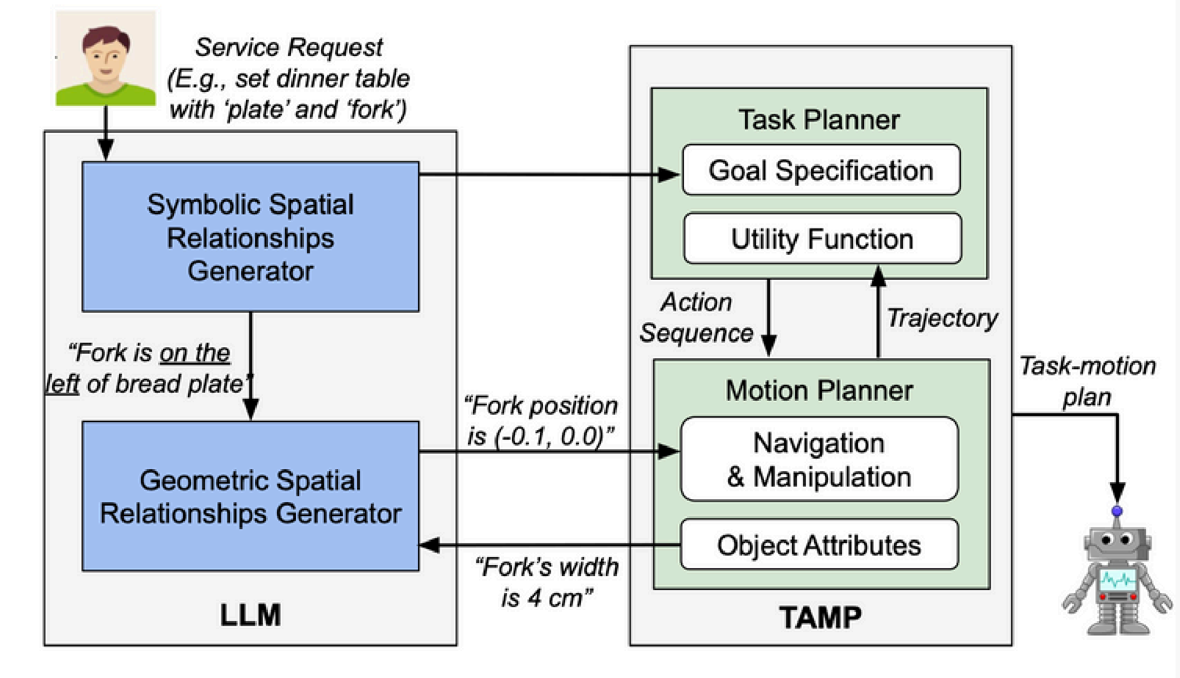


Fig. 2: LLM-GROP takes service requests from humans for setting tables and produces a task-motion plan that the robot can execute. LLM-GROP is comprised of two key components: the LLM and the Task and Motion Planner. The LLM is responsible for creating both symbolic and geometric spatial relationships between the tableware objects. This provides the necessary context for the robot to understand how the objects should be arranged on the table. The Task and Motion Planner generates the optimal plan for the robot to execute based on the information provided by the LLM.

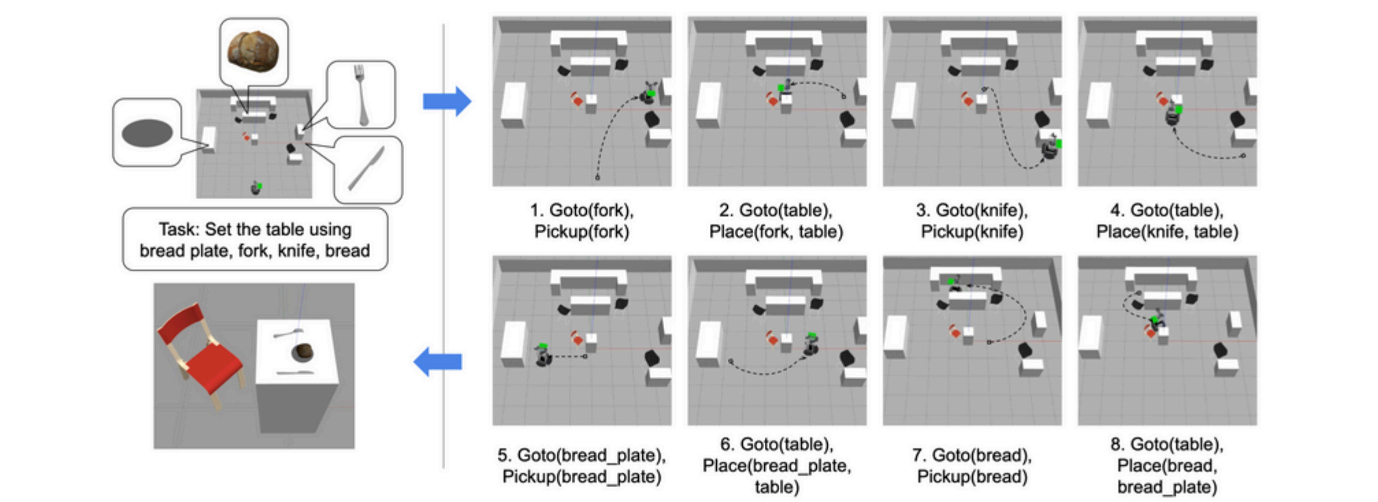


Fig. 3: An illustrative example of LLM-GROP showing the robot navigation trajectories (dashed lines) as applied to the task of "set the table with a bread plate, a fork, a knife, and a bread." LLM-GROP is able to adapt to complex environments, using commonsense extracted from GPT-3 to generate efficient (i.e., minimize the overall navigation cost) and feasible (i.e., select an available side of the table to unload) pick-and-place motion plans for the robot.

DING, Yan, ZHANG, Xiaohan, PAXTON, Chris, et al. Task and motion planning with large language models for object rearrangement. arXiv preprint arXiv:2303.06247, 2023. [link], [code]

Reflexion: Language Agents with Verbal Reinforcement Learning [Shinn+ (Northeastern Univ.), NeurIPS23]

- **Objective**: Develop a method for reinforcing language agents using verbal feedback instead of traditional weight updates. Reflexion focuses on improving decision-making in agents through reflective text stored in episodic memory.
- **Methodology**: Reflexion agents verbally reflect on task feedback, maintaining a reflective text in an episodic memory buffer. This method aims to enhance decision-making in subsequent trials by incorporating different types of feedback (scalar values or free-form language).
- **Key Innovation:** Reflexion represents a shift from conventional reinforcement learning, using linguistic feedback to direct agents towards improved performance. It has shown significant improvements across diverse tasks, including sequential decision-making, coding, and language reasoning, achieving notable accuracy in benchmarks like the HumanEval coding task.

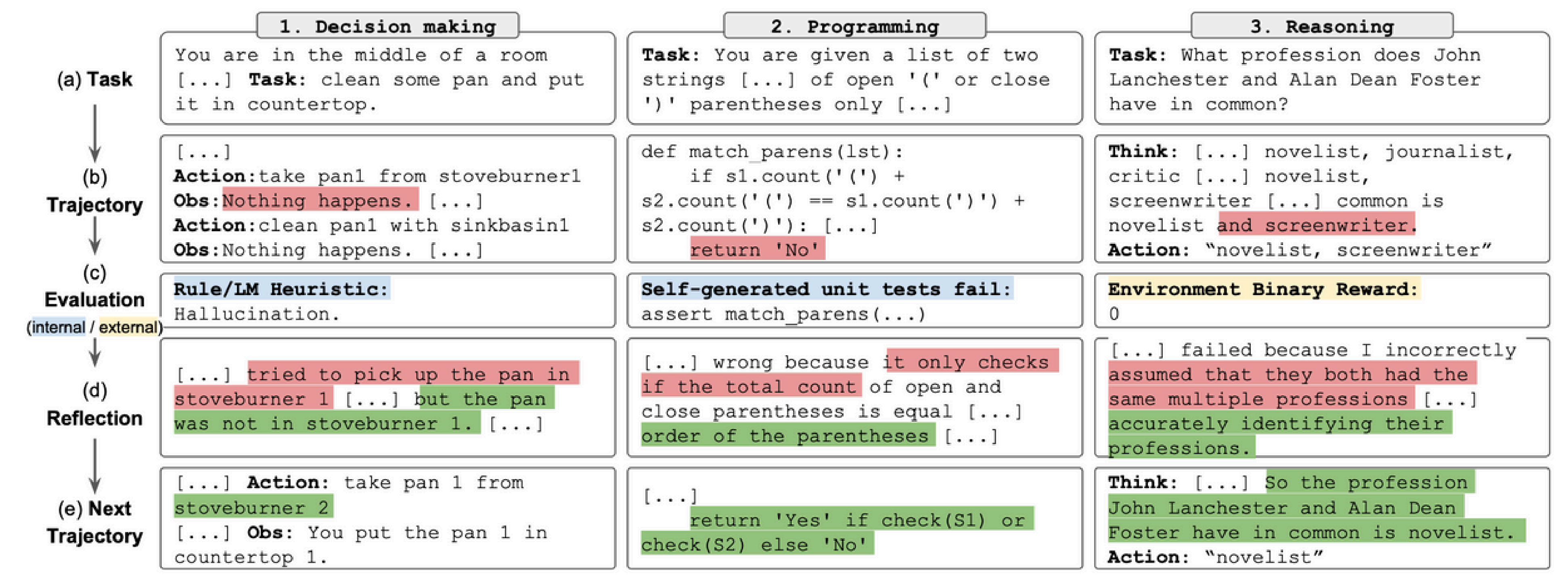
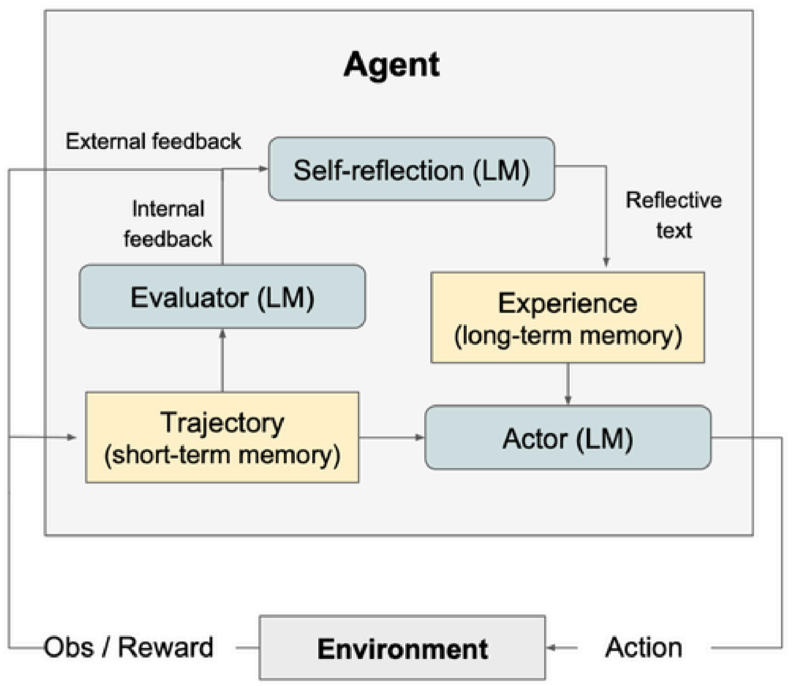


Figure 1: Reflexion works on decision-making 4.1, programming 4.3, and reasoning 4.2 tasks.



Algorithm 1 Reinforcement via self-reflection

Initialize Actor, Evaluator, Self-Reflection: M_a, M_e, M_{sr}
Initialize policy $\pi_\theta(a_i|s_i)$, $\theta = \{M_a, mem\}$
Generate initial trajectory using π_θ
Evaluate τ_0 using M_e
Generate initial self-reflection sr_0 using M_{sr}
Set $mem \leftarrow [sr_0]$
Set $t = 0$
while M_e not pass or $t <$ max trials **do**
  Generate $\tau_t = [a_0, o_0, \ldots a_i, o_i]$ using π_θ
  Evaluate τ_t using M_e
  Generate self-reflection sr_t using M_{sr}
  Append sr_t to mem
  Increment t
end while
return

Figure 2: (a) Diagram of Reflexion. (b) Reflexion reinforcement algorithm

SHINN, Noah, CASSANO, Federico, GOPINATH, Ashwin, et al.Reflexion: Language agents with verbal reinforcement learning. In : Thirty-seventh Conference on Neural Information Processing Systems. 2023. link, code