# Reinforcement Learning with Pretrained Models

Nov. 2022

Dr. Honguk Woo

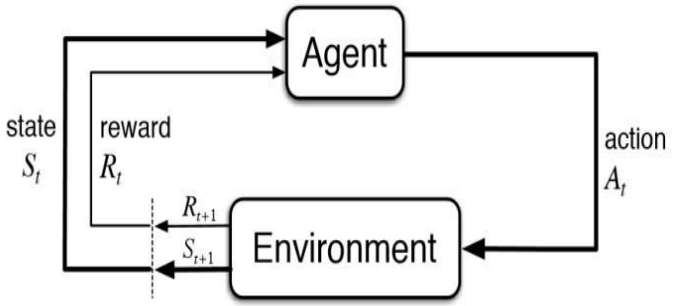
Department of Computer Science and Engineering, SKKU

# CSI Lab @ SKKU

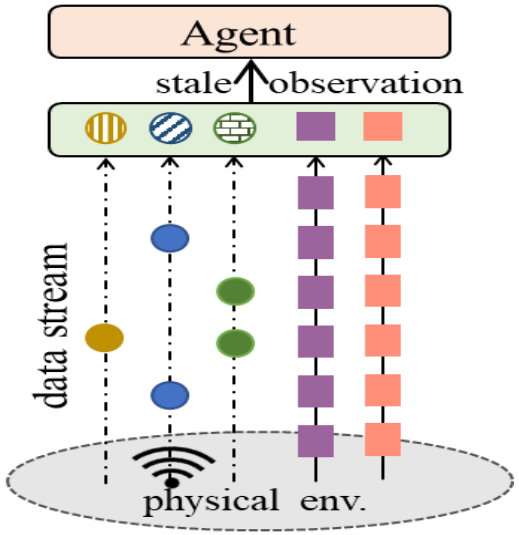
- Complex sequential decision tasks :
  - How to apply Reinforcement, Multi-task, Meta Learning
  - Lab2Real : adapt models for real-world settings

| In the Lab | Real-world |
|---|---|
| Small scale | Large scale |
| Specific systems | Various systems |
| Stationary env. | Non-stationary env. |
| Given (seen) dataset | Unseen dataset |
| No delay | Delayed data |
| Sufficient resource | Constrained resource |

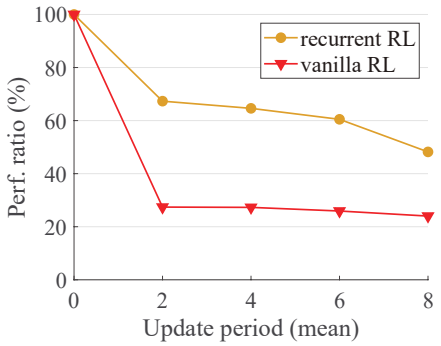
*An agent learns through interaction with the environment*




*An agent makes inferences on real-time data*

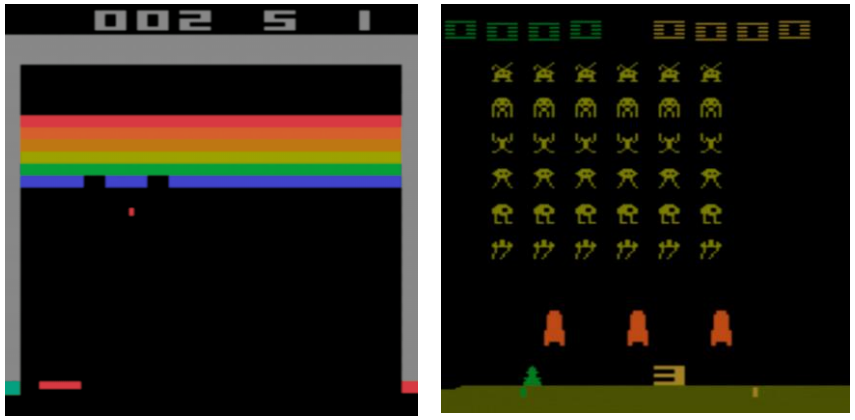



*Data delays degrade ML performance in real-world*

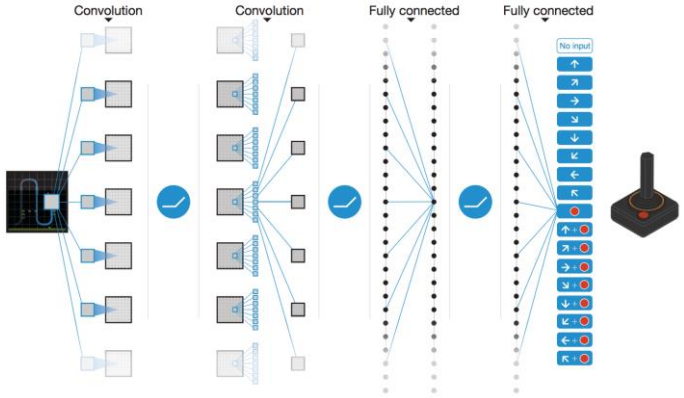



Images from Giphy

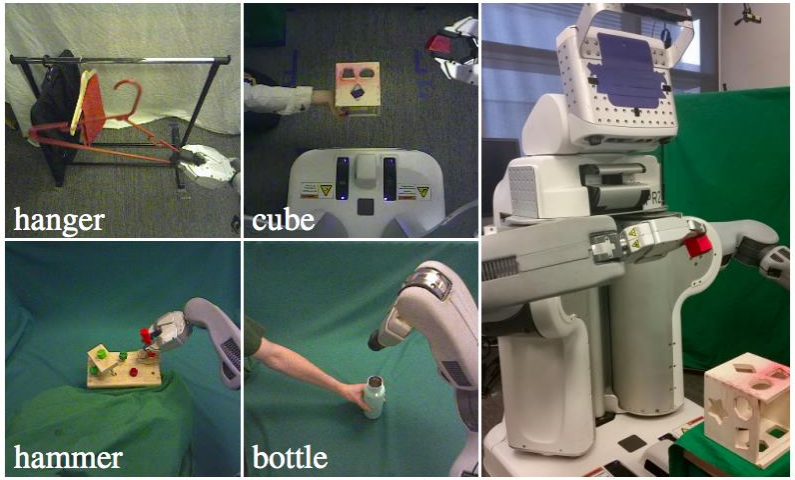
# Reinforcement Learning



Mnih et al., Atari game AI, *Nature* (2015)




Pieter Abbeel et al., Autonomous helicopter flight

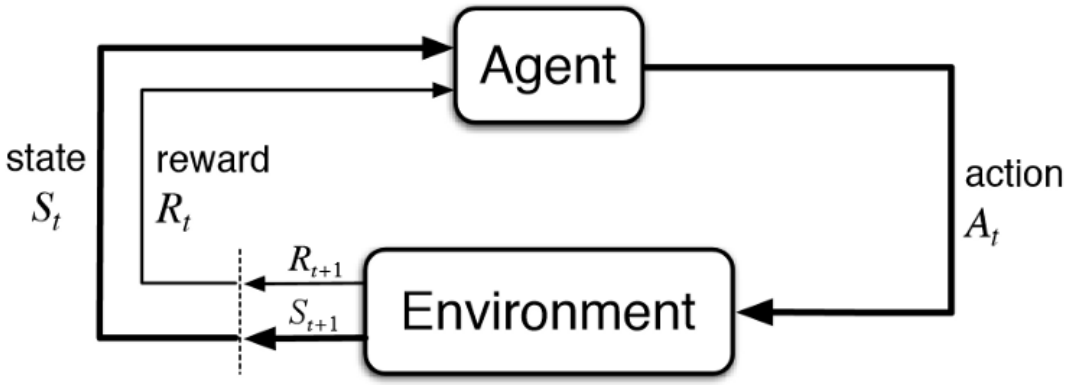



Sergey Levine et al., Deep Visuomotor Policies

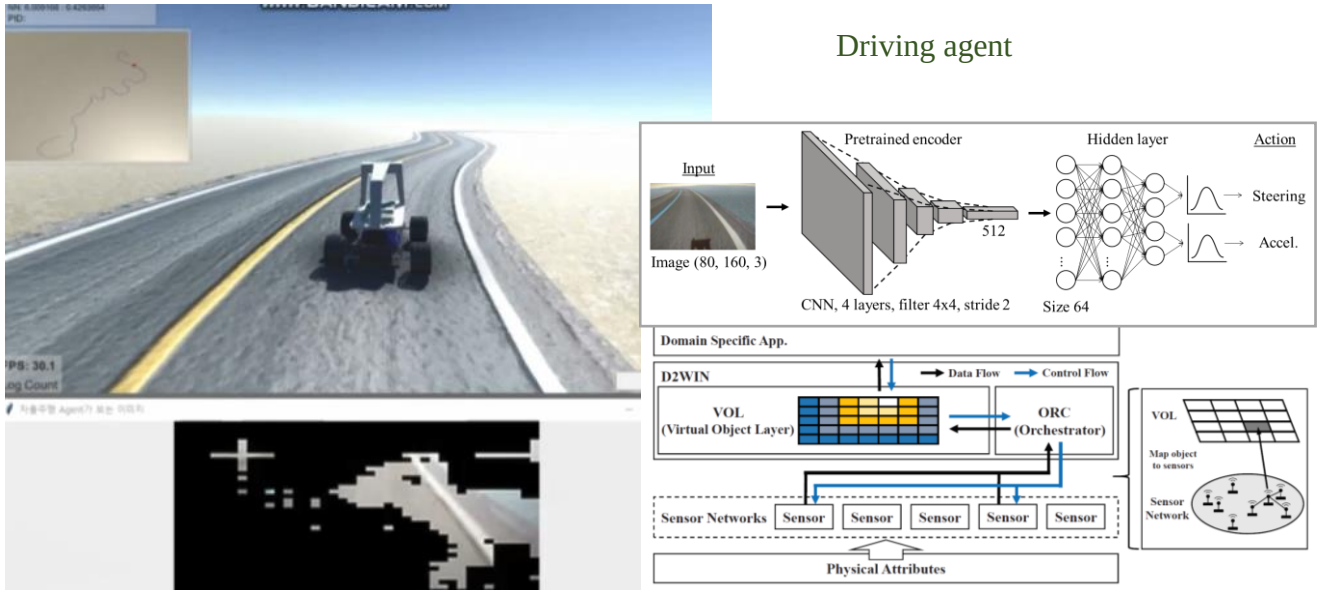
Naver, Human friendly navigation

# Reinforcement Learning

- Learning by continuous interactions between an agent and an environment
  - Interaction : State (Observation) – Action – Next State & Reward
  - Algorithm framework designed to train an agent on a task through interactions with the environment




Ref. https://towardsdatascience.com/




Driving agent

Donkey simulation test

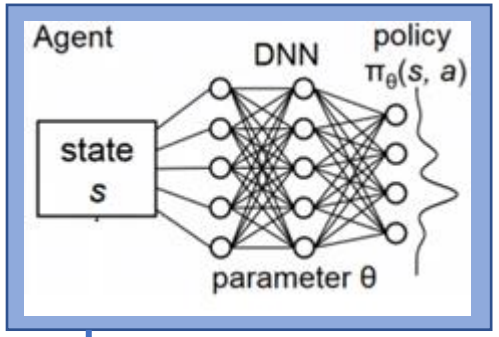
# Reinforcement Learning

- Mathematical formulation for reinforcement learning problems
  - Markov Decision Process (**MDP**) : $(S, A, R, P)$
    - A set of **States** $S$
    - A set of **Actions** $A$ - possible actions that agent can make in the environment
    - **Transition** probability $P$ – distribution over next state given state-action
    - Distribution of **Reward** $R$ given state-action
      - Feedback measuring success of failure of actions
      - Discount factor $\gamma$

$$R_t = \sum_{i=t} r_i = r_t + r_{t+1} \dots + r_{t+n} + \cdots \qquad R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

- Agent learns **policy** $\pi$
  - Policy : function from **States** to **Actions**
  - Maximizing future rewards over time
  - Solving the MDP problem

**Policy**





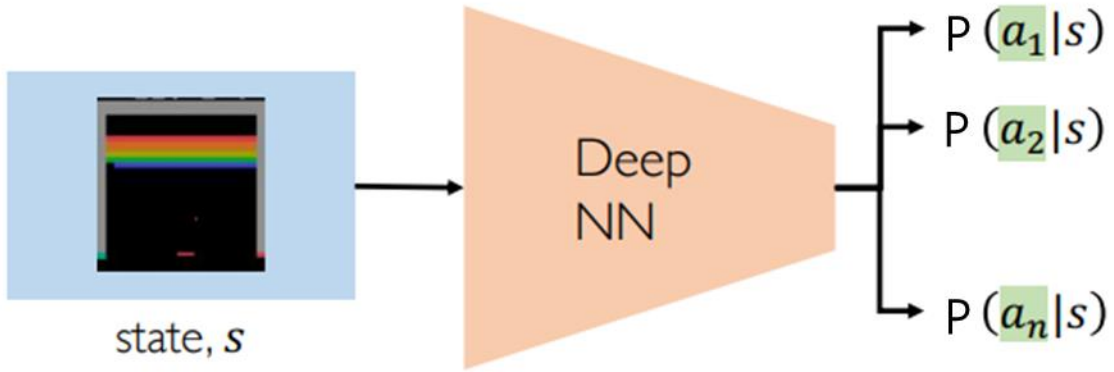
- Mathematical formulation for reinforcement learning problems
  - Markov Decision Process (MDP) : $(S, A, R, P)$
    - A set of **States** $S$
    - A set of **Actions** $A$ - possible actions that agent can make in the environment
    - **Transition** probability $P$ – distribution over next state given state-action
    - Distribution of **Reward** $R$ given state-action
      - Feedback measuring success of failure of actions
      - Discount factor $\gamma$

- Agent learns policy $\pi$
  - Policy : function from **States** to **Actions**
  - Maximizing future rewards over time
  - Solving the MDP problem

# Reinforcement Learning

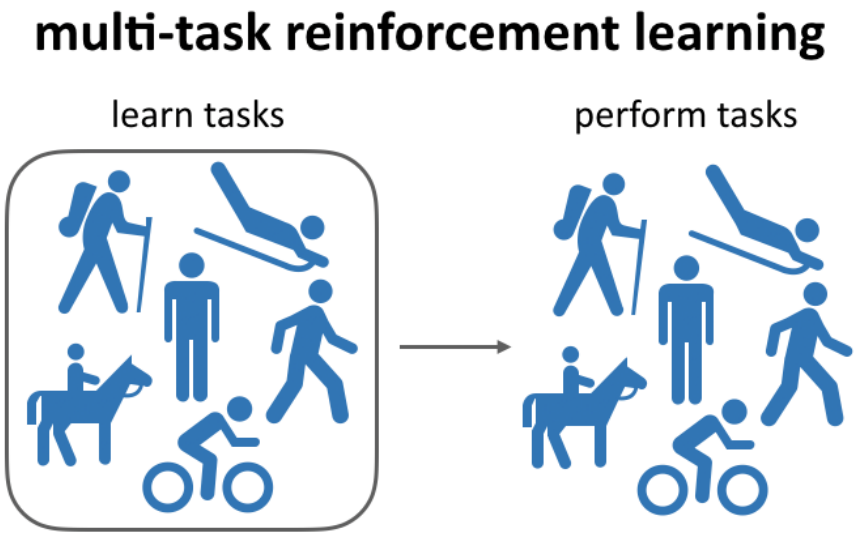
- Policy-based reinforcement learning
  - (*directly*) Optimize policy $\pi(s)$ using **Policy Gradient** method
  - Policy : function from states to actions maximizing discounted cumulative rewards; can be stochastic
  - Optimal policy $\pi : S \times A \rightarrow [0,1]$ *s.t.* $argmax_{\pi}\, \mathbb{E}_{a_t \sim \pi}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$

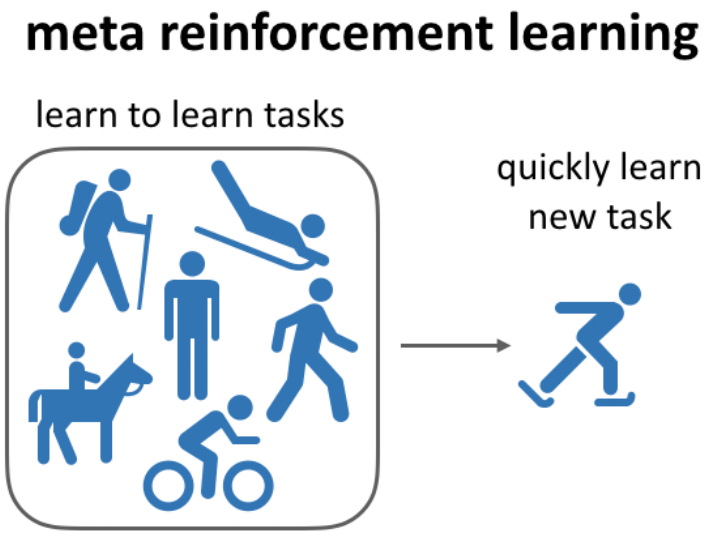
$\pi(s) \sim P(action/state)$




Ref. MIT 6.s191 deep learning

# Multi-task, Meta Learning in the RL context




https://meta-world.github.io/

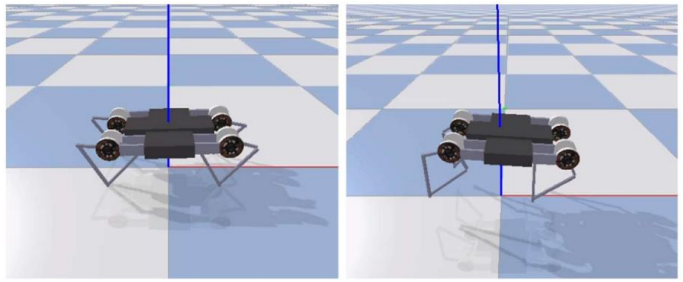
# How to solve complex sequential decision problems in the multi-task, RL context


Structure Learning-Based Task Decomposition For
Reinforcement Learning In Non-Stationary Environments (AAAI 2022)

Honguk Woo, Gwangpyo Yoo, Minjong Yoo
Department of Computer Science and Engineering, Sungkyunkwan University

# Agents in Non-stationary env.

- Non-stationary: environments where tasks vary over time
  - e.g., drone in windy environments  minitaur with dynamic weights

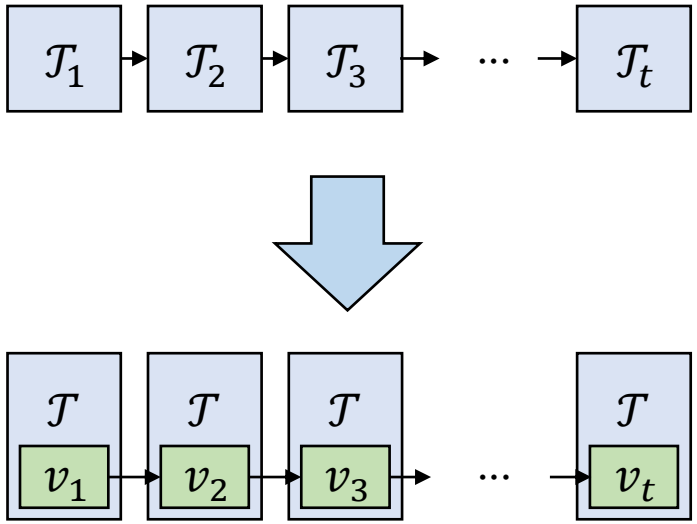


- Reinforcement learning in a non-stationary environment
  - Since an RL objective should be governed by varying tasks, the objective accordingly changes during learning

# Agents in Non-stationary env.

- Non-stationary environment problem

- We define one task as Markov Decision Process (MDP)
- Since tasks change over time, a non-stationary environment is represented by a sequence of MDPs
- By defining a hidden parameter $v_t$ that changes, we assume that the transition probability and reward function change with $v_t$
- So, we can solve the non-stationary environment problem by inferring $v_t$ representing the tasks and optimizing the policy based on $v_t$




<reformulate non-stationary environment problem>

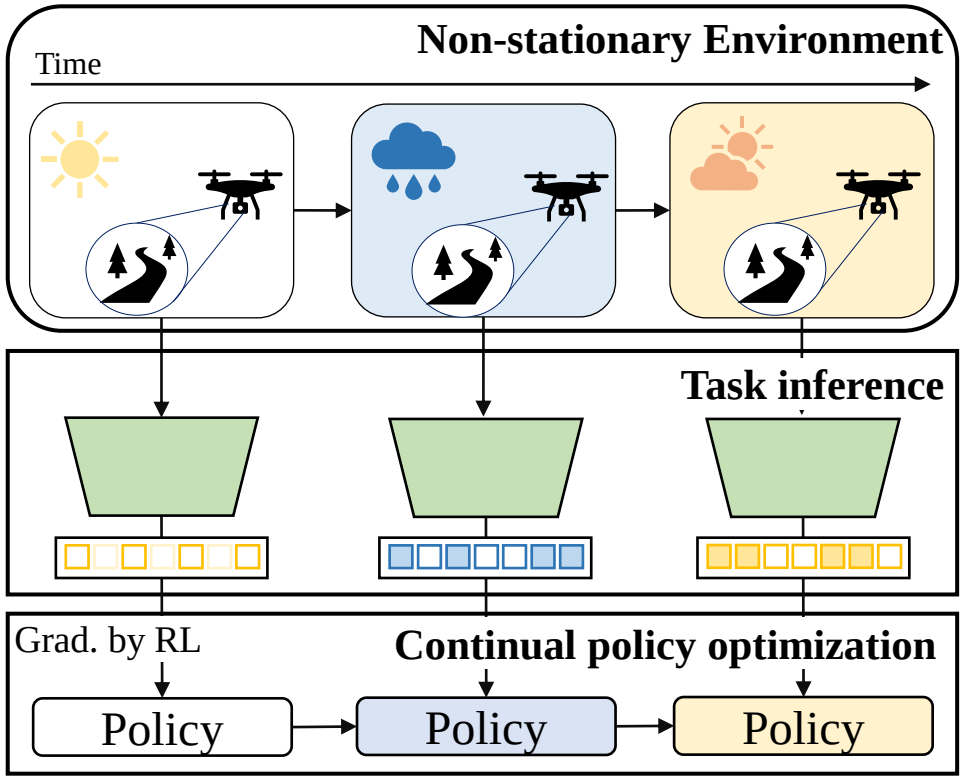
# Our approach : task inference + continual optimization

## 1. Task inference in a non-stationary environment

- Embed varying tasks and use them for learning policies
- How can we effectively embed time-varying parameters that represent tasks?

## 2. Continual policy optimization in a non-stationary environment

- Continuously learn policies using the task embedding in an environment where tasks change
- How can we solve the ‘catastrophic forgetting’ problem of disrupting the policy parameters for the knowledge on previous tasks?



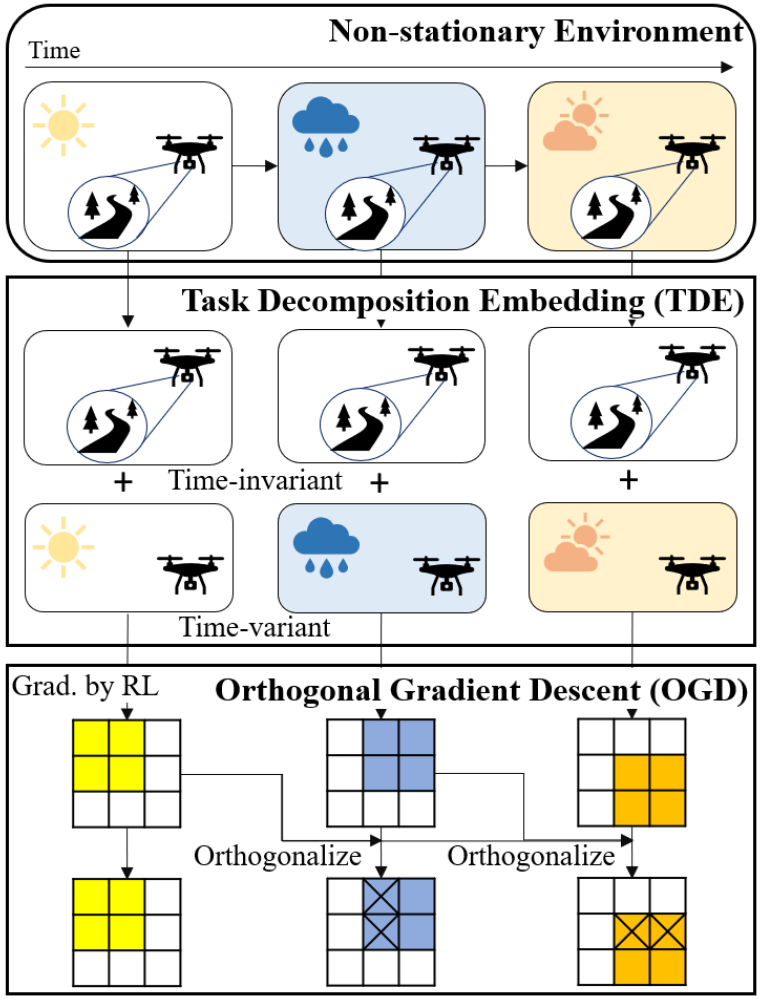
# Our approach : task inference + continual optimization

## 1. Task inference: Task decomposition embedding (TDE)

- Time-variant parameters are entangled in the different parts of tasks
- Decompose tasks through **CycleGAN-based structural learning** to extracting time-variant parameters

## 2. Continual policy optimization: Sliding-windowed orthogonal gradient descent (OGD)

- The overlapping direction of the gradients incurs ‘catastrophic forgetting’
- Remove the interference part of the gradients through **orthogonalization**



# Task Decomposition Embedding

- Objective

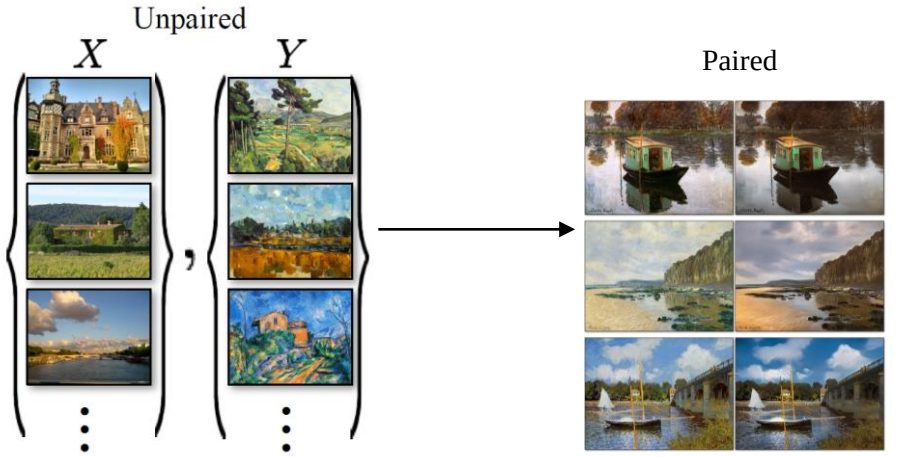
- Embed time-variant parameters in a non-stationary environment
- Decompose tasks into a time-variant part and a time-invariant part

- Procedure

1. Decompose tasks and generate transition pairs using CycleGAN
2. Learning the task embedding using auto-encoder

[CycleGAN]

- Create data pairs from datasets with different characteristics
- Find out common features and discriminate different features




Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *ICCV* 2017
Gavranović, Bruno. "Learning Functors using Gradient Descent." *arXiv* 2020

# Task Decomposition Embedding

## 1. Decompose tasks and generate transition pairs using CycleGAN

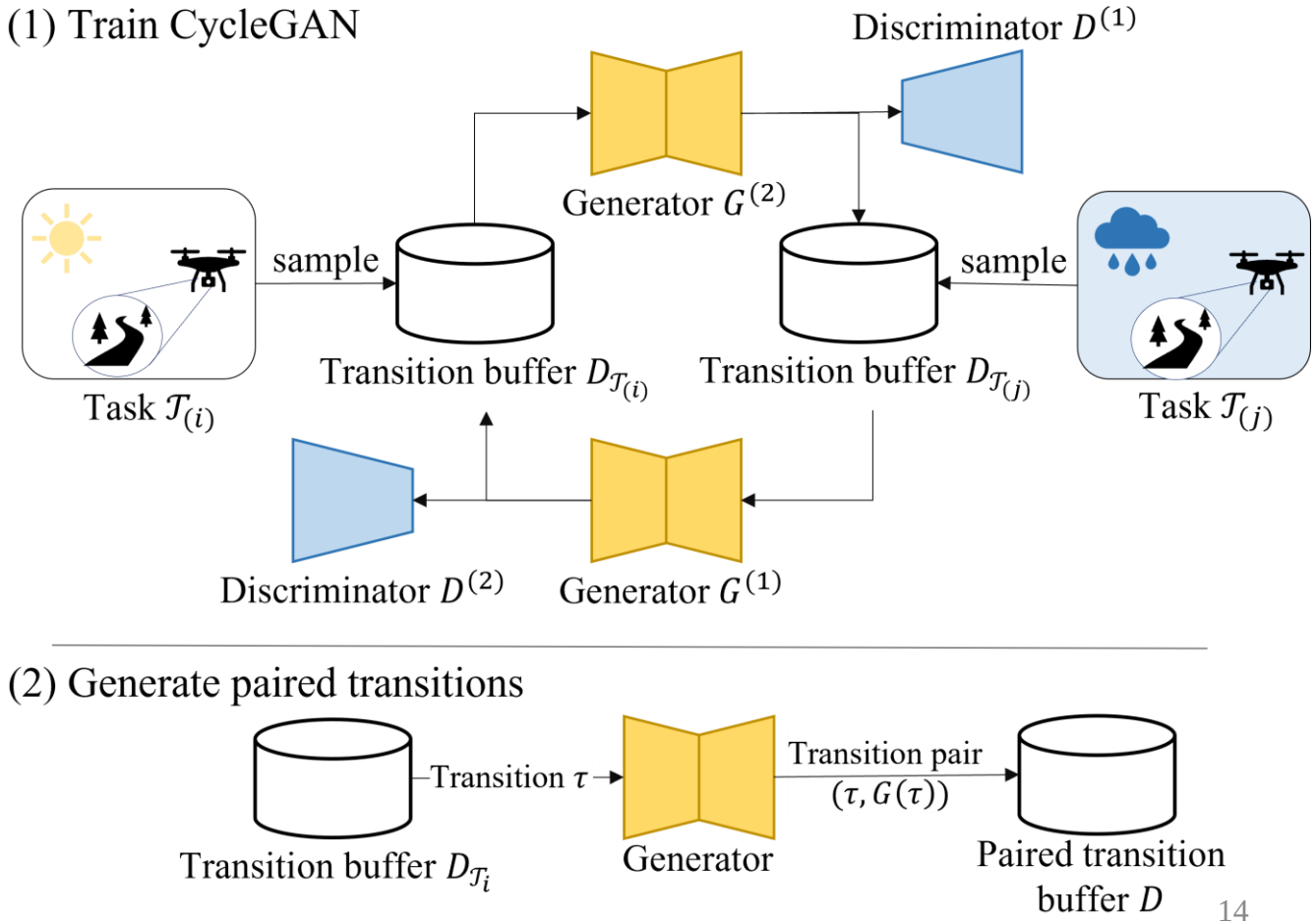
Loss function

- Adversarial loss

$$\mathcal{L}_{gan}(G, D) = \mathbb{E}_{y \sim \mathcal{D}_{\mathcal{T}_{(1)}}}[\log(D(y)] + \mathbb{E}_{x \sim \mathcal{D}_{\mathcal{T}_{(2)}}}[\log(1 - D(G(x))].$$

- Cycle consistency loss

$$\mathcal{L}_{con}(G^{(1)}, G^{(2)}) = \mathbb{E}_{x \sim \mathcal{D}_{\mathcal{T}_{(2)}}}[||G^{(2)} \circ G^{(1)}(x) - x||_1] + \mathbb{E}_{y \sim \mathcal{D}_{\mathcal{T}_{(1)}}}[||G^{(1)} \circ G^{(2)}(y) - y||_1].$$



# Task Decomposition Embedding

## 2. Learning the task embedding using auto-encoder

Loss function

- Reconstruction loss for $u$

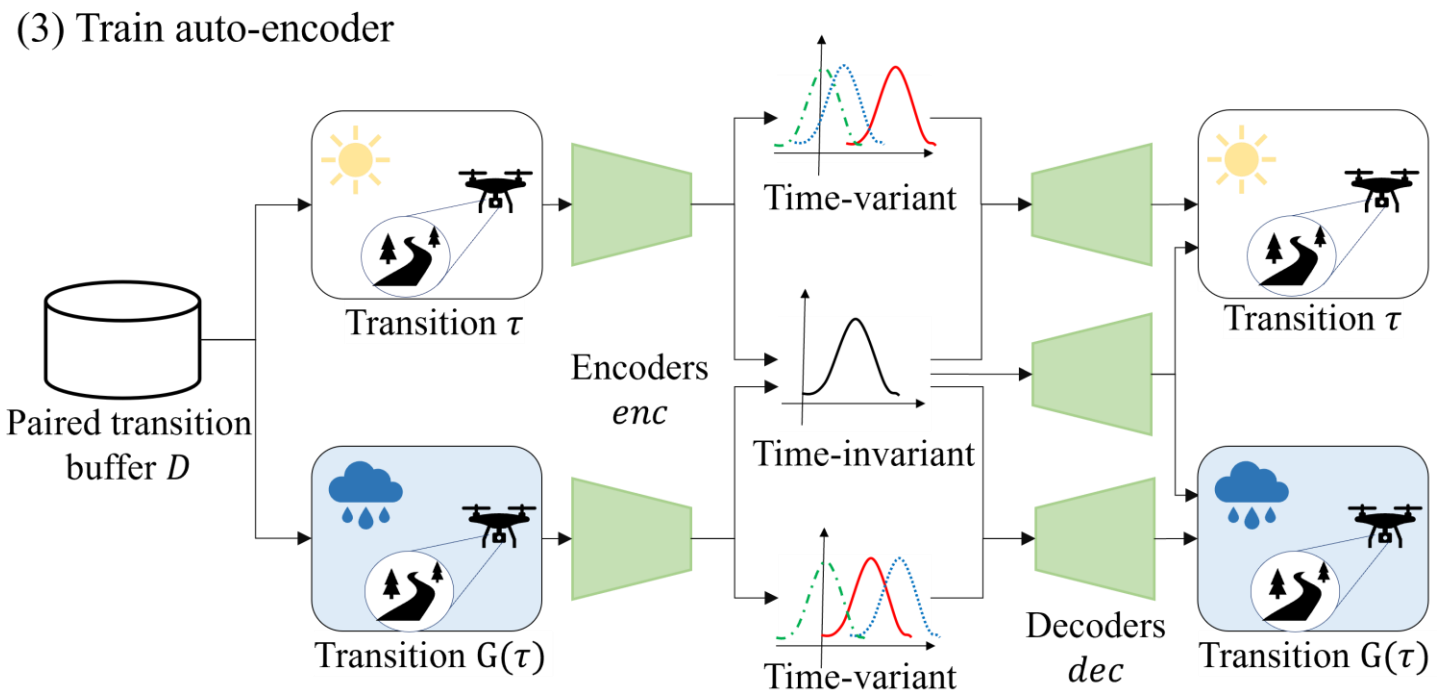
$$\mathcal{L}_{recon}(enc, dec) = \|\tau - dec(\mathbf{u}, \mathbf{0})\|_2$$

- Similarity loss

$$\mathcal{L}_{sim}(enc) = ||\mathbf{u} - \mathbf{u}'||_2 = ||enc(\tau)|_{\mathcal{U}} - enc \circ G(\tau)|_{\mathcal{U}}||_2$$

- Basic reconstruction loss

$$\mathcal{L}_{ae}(enc, dec) = \|\tau - dec \circ enc(\tau)\|_2$$



# Sliding-windowed orthogonal gradient descent

- Objective

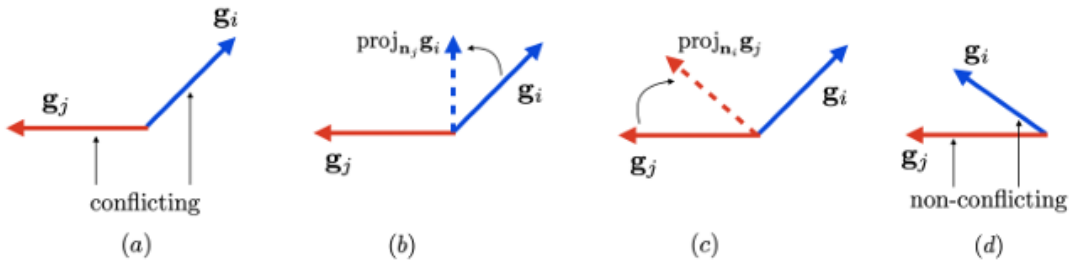
- Train the policy without forgetting the knowledge of the previous task
- Remove the interference between the previous gradient and the generated gradient

- Procedure

1. Generate gradients through reinforcement learning algorithm
2. Orthogonalize the generated gradient to the gradient up to certain previous time

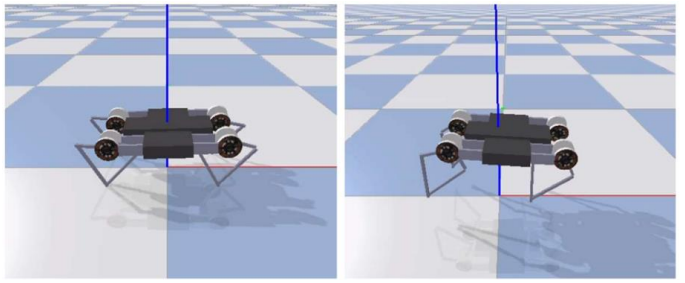
[Gradient Surgery]

- Adjust the gradient generated during learning for multi-task reinforcement learning
- Claim that the gradient conflict problem degrades learning performance
- Remove the part where gradients for different tasks cause conflicts



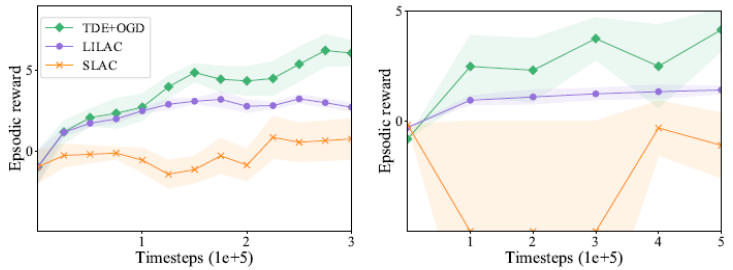

1) Yu, Tianhe, et al. "Gradient surgery for multi-task learning." *arXiv* 2020

# Evaluation : simulation



- Comparison with state-of-the-art

- In 2-d navigation and minitaur environment, TDE+OGD shows 46.4% improvement on average over LILAC(1)

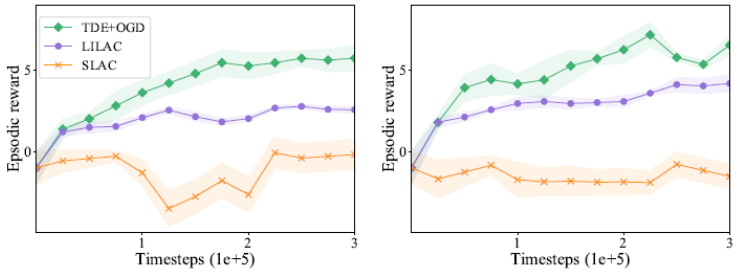



(a) 2-d navigation (b) Minitaur

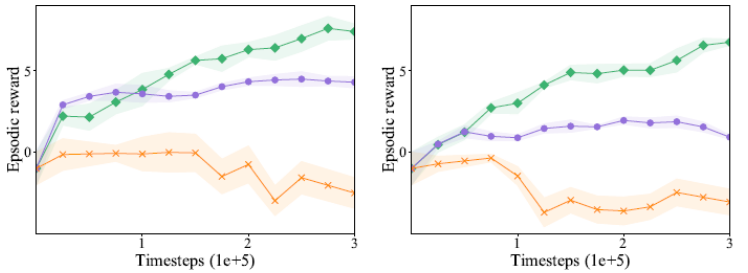
- Comparison with task dynamics and uncertainty

- Both TDE+OGD and LILAC achieve stable performance, showing no significant performance degradation between low and high dynamics.

- For task uncertainty TDE+OGD shows performance drop of 17.3% on average for high uncertainty compared to low uncertainty, whereas LILAC shows a performance drop of 62.1%




(a) Low dynamics (b) High dynamics




(c) Low uncertainty (d) High uncertainty

Xie, Annie, James Harrison, and Chelsea Finn. "Deep reinforcement learning amidst lifelong non-stationarity." *arXiv* 2020

# Evaluation: AIRSIM

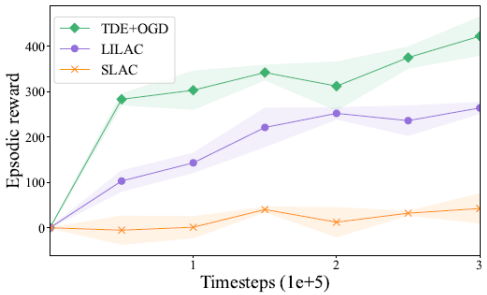
- Comparison with state-of-the-art

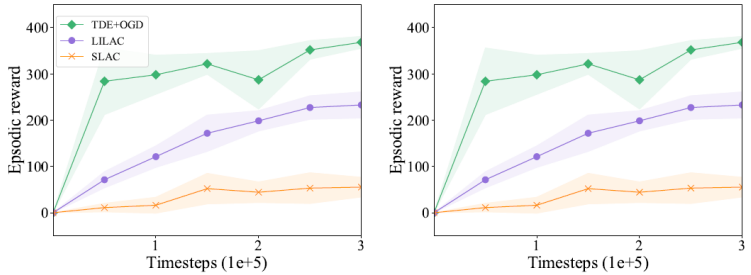
- In the Airsim case study, TDE+OGD shows 67.4% improvement on average over LILAC

- Performance w.r.t. task dynamics and uncertainty

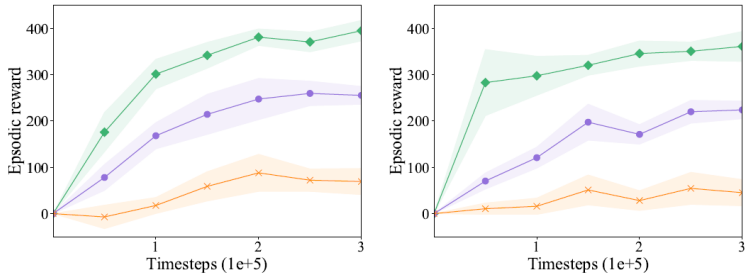
- Similar to the simulation experiment, showing no significant performance degradation between low and high dynamics.

- For task uncertainty, TDE+OGD shows a performance drop of 8.6% on average for high uncertainty compared to low uncertainty, whereas LILAC shows a performance drop of 12.28%




(a) Low dynamics (b) High dynamics




(c) Low uncertainty (d) High uncertainty

# How to solve complex sequential decision problems in the “pretrained” multi-task, RL context

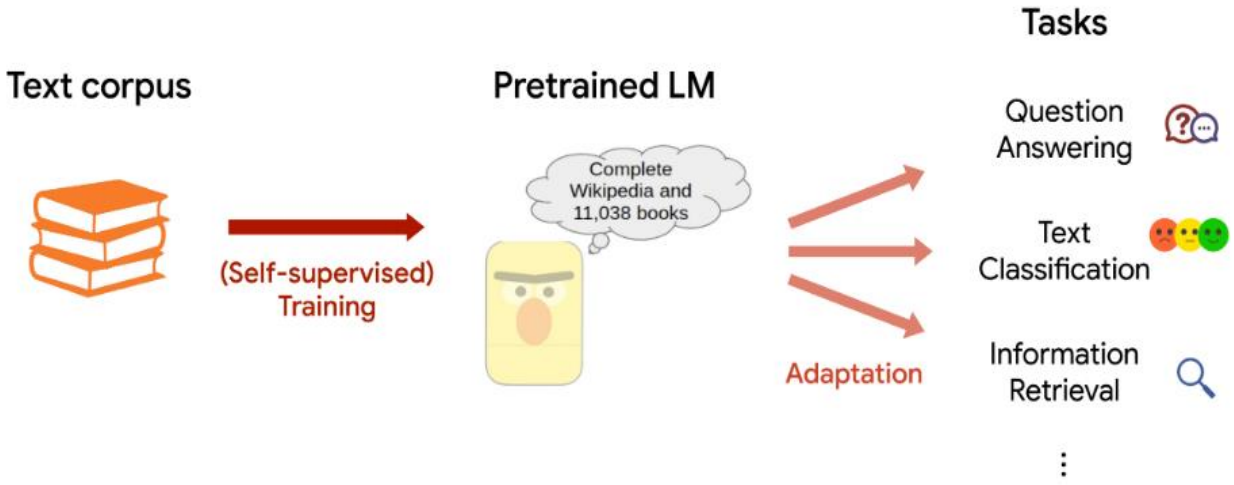

Skills Regularized Task Decomposition for Multi-task Offline Reinforcement Learning (NeurIPS 2022)

Minjong Yoo, Sangwoo Cho, Honguk Woo
Department of Computer Science and Engineering, Sungkyunkwan University

# Pretrained Models for language, vision

- Huge success in deep learning
  - From hand-crafted features to automatic learning; but data-intensive tasks with poor generalization ability
- So, transfer learning with pretraining and finetuning

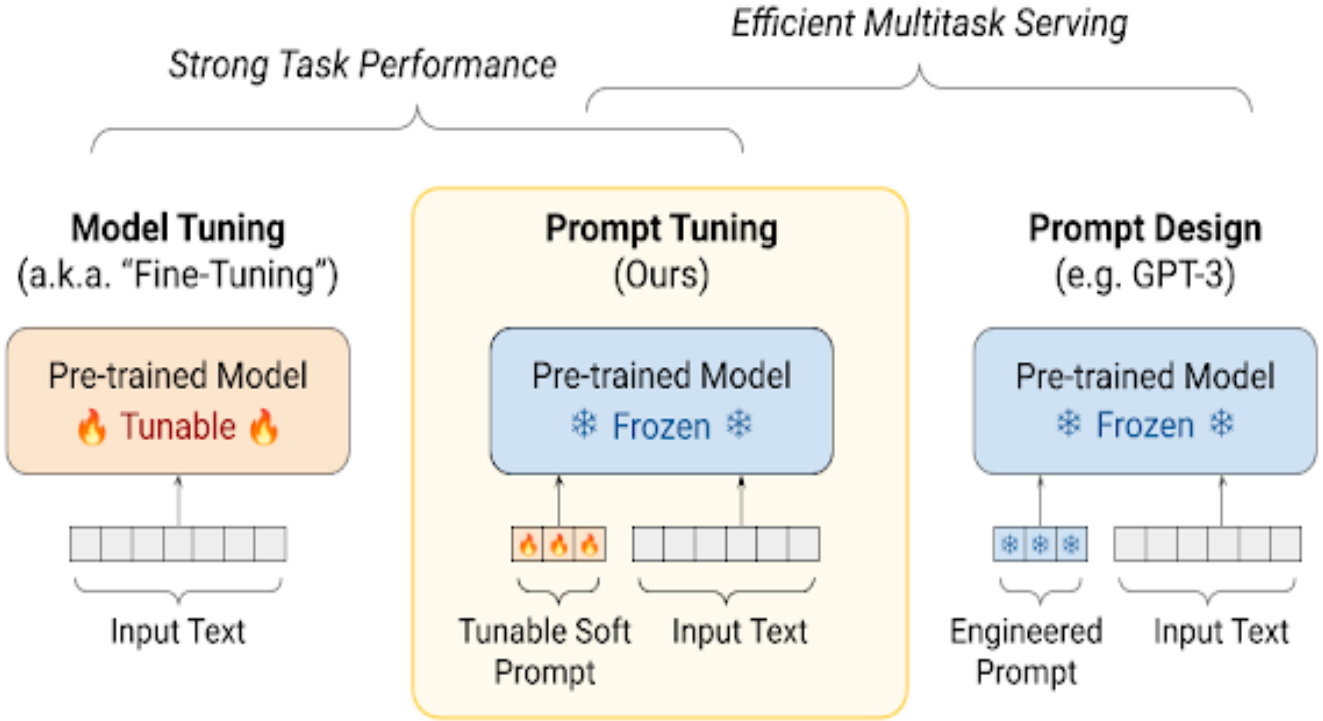
- Pretrained models, e.g.,
  - Language models : GPT, Bert
  - Vision models : ResNet
  - Vision-language models : Clip




http://ai.stanford.edu/blog/linkbert/

# Pretrained Models

- Finetuning for downstream tasks




https://ai.googleblog.com/2022/02/guiding-frozen-language-models-with.html

# Vision-Language Pretrained Models

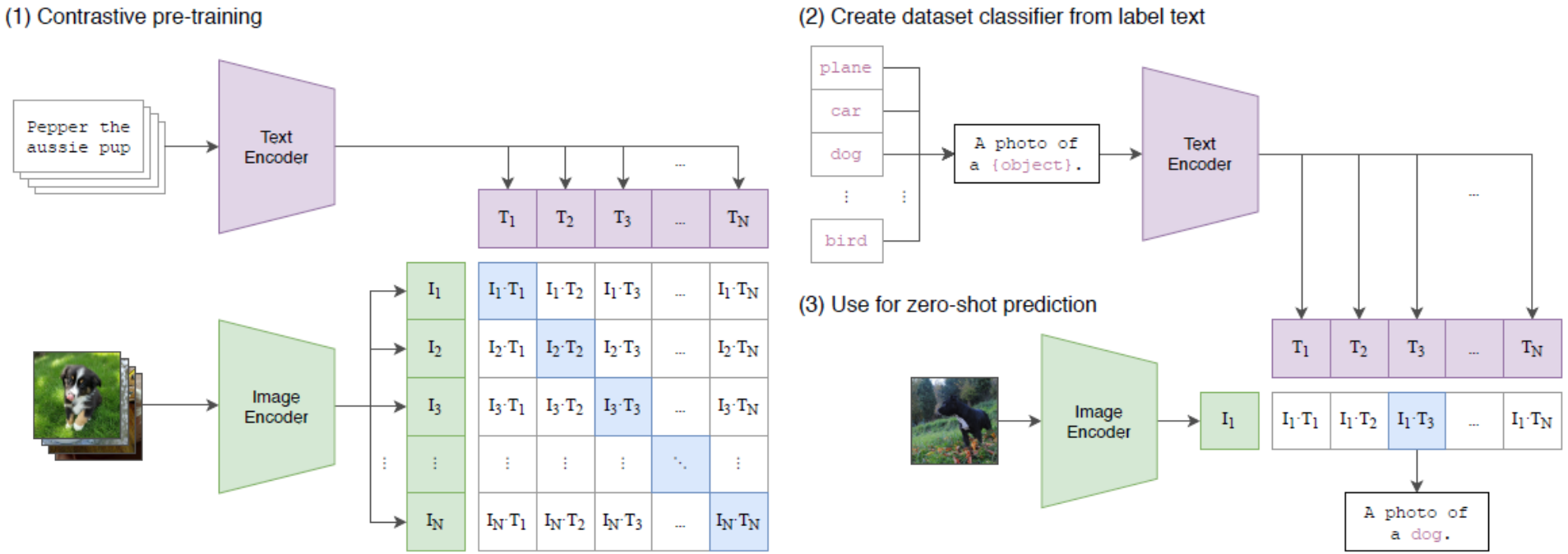
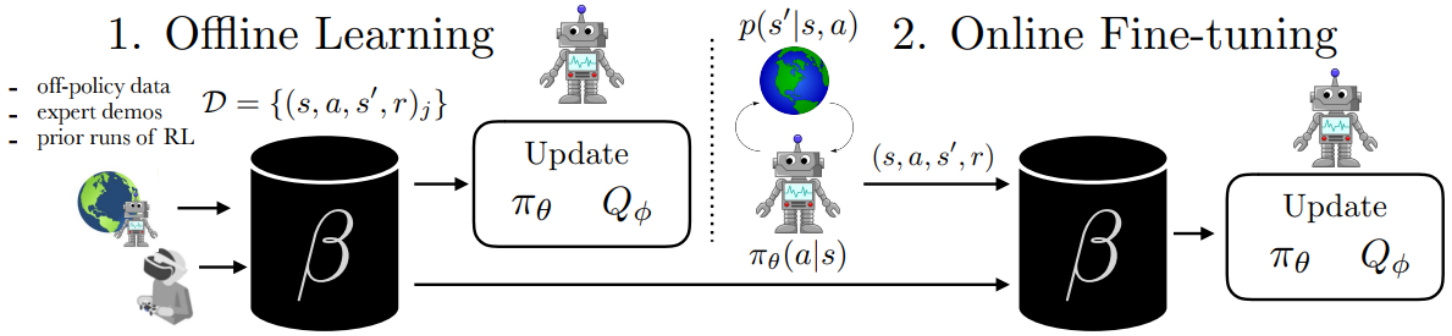



*Figure 1.* Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Learning Transferable Visual Models From Natural Language Supervision (OpenAI, ICML 2021)

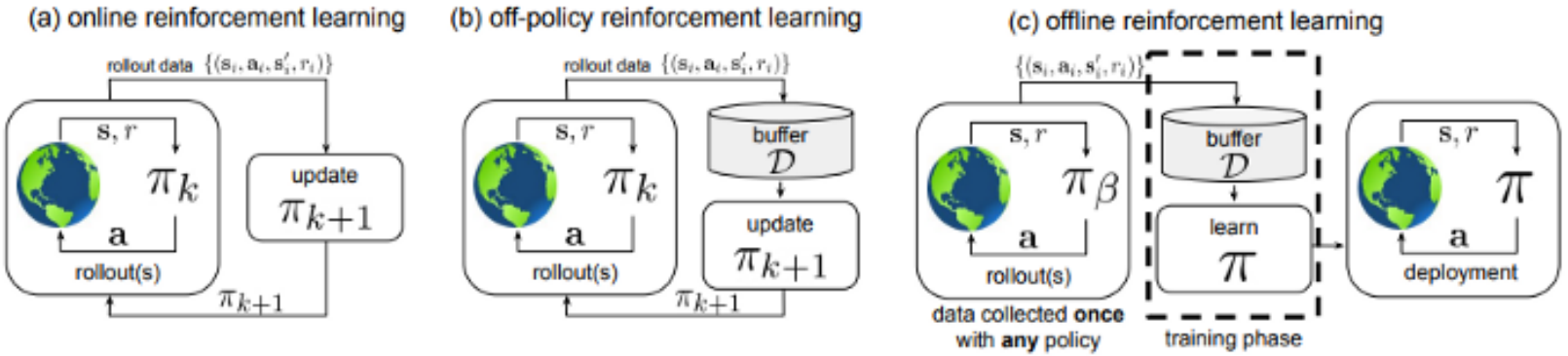
# Toward Pretrained Models for RL tasks

- Pretrained models and finetuning




AWAC: Accelerating Online Reinforcement Learning with Offline Datasets (UC Berkeley, 2020)

- From online learning to offline, data-driven learning

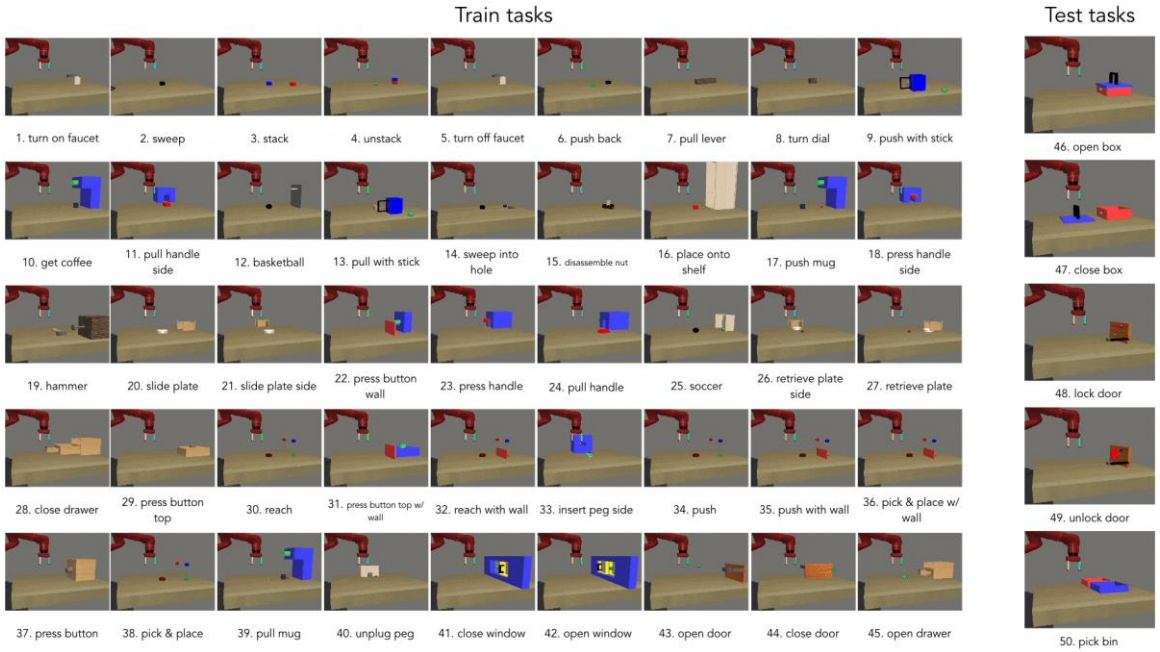


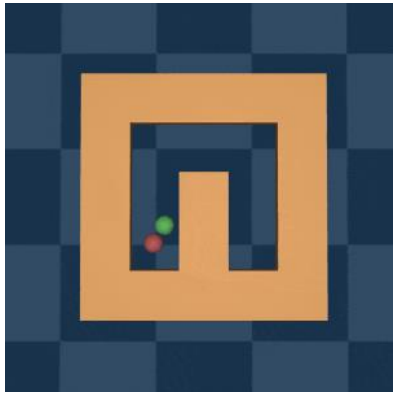

Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems

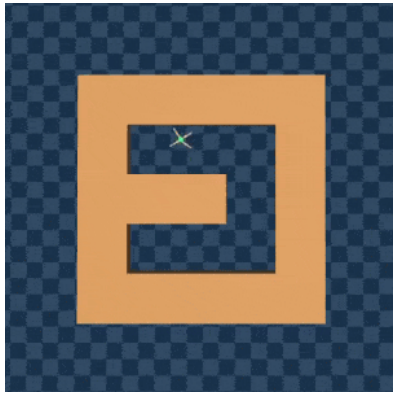
# Offline datasets and RL tasks

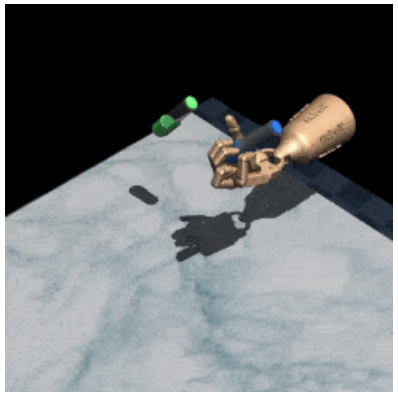
- D4RL : Datasets for Deep Data-Driven Reinforcement Learning (UC Berkeley, Google AI Research)

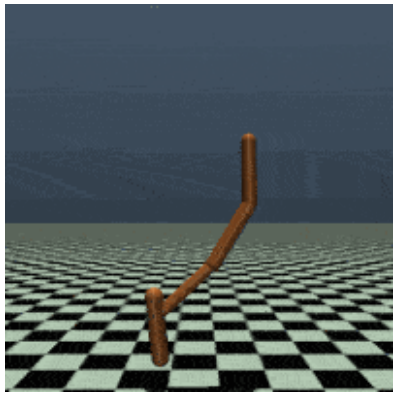
- Metaworld : 50 manipulation tasks for multi-task, meta learning

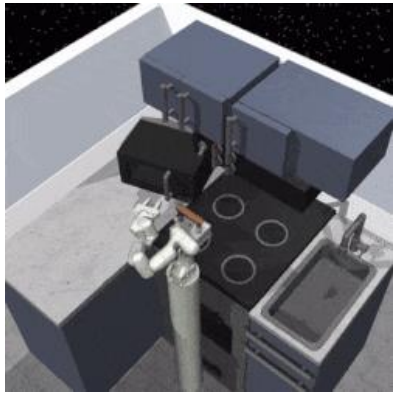




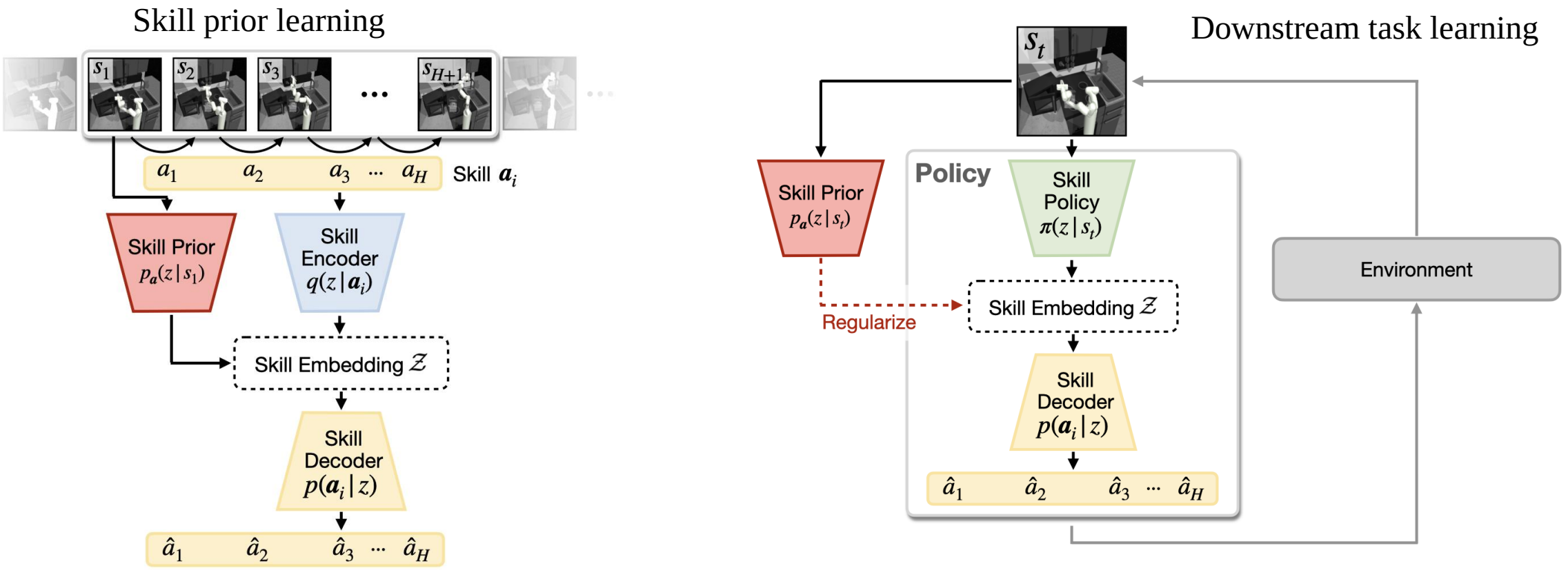






# Skills as a Pretrained Model

- Skills pretrained models (prior) and finetuning (posterior training)
  - Skills are extracted from task-agnostic offline datasets




Accelerating Reinforcement Learning with Learned Skill Priors (CoRL, 2020)

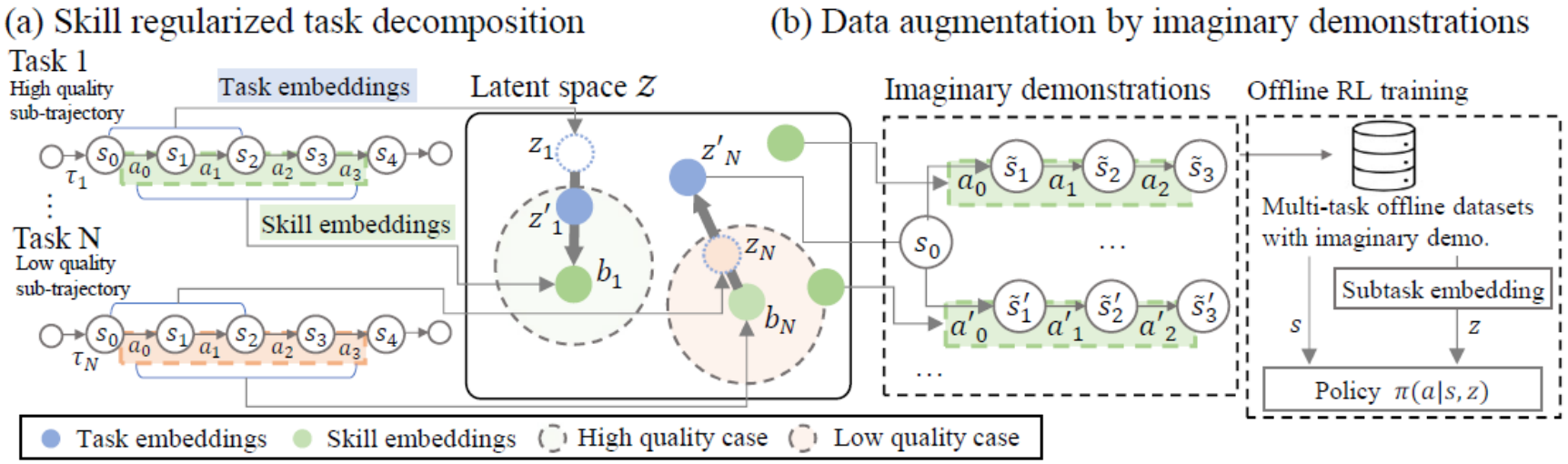
# Our approach : skill regularized + imaginary demo.

## 1. Skill regularized task decomposition

- To address different-quality and heterogenous datasets in the context of multi-task offline RL
- Decompose an individual task into achievable subtasks through the quality-aware joint learning on skills and tasks

## 2. Data augmentation by imaginary demonstrations

- To use augmented data that are likely to be generated by an expert policy
- Create trajectories with the skill decoder that generates actions and the task decoder that models transition probability and reward function




(NeurIPS 2022)

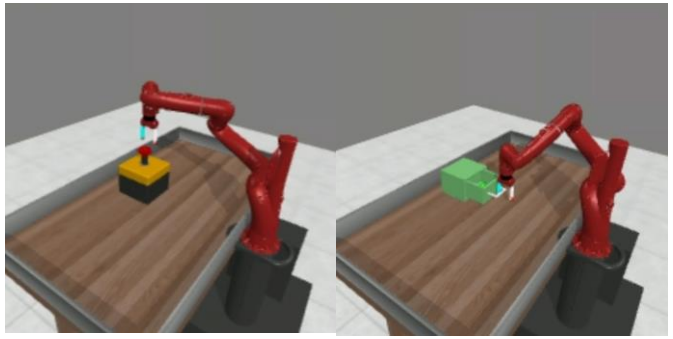
# Results: Simulation Environment

- Meta-world environment (MT10)

- MT10: 10 different robot-arm manipulation environments such as push, pick and place

- Experiment results

- Our model (SRTD, SRTD+ID) shows 8.67%~17.67% improvement over other baselines (SoftMod, PCGrad) in offline RL settings



| Datasets | | | Comparison | | | Our model | |
|---|---|---|---|---|---|---|---|
| MR | RP | ME | TD3+BC | PCGrad | SoftMod | SRTD | SRTD+ID |
| 10 | 0 | 0 | 19.73 ± 2.71% | 20.66 ± 4.27% | 13.43 ± 2.67% | 21.24 ± 1.40% | **23.87 ± 2.22%** |
| 0 | 10 | 0 | 25.93 ± 5.91% | 27.31 ± 3.15% | 29.04 ± 0.58% | 38.97 ± 3.38% | **41.91 ± 5.88%** |
| 0 | 0 | 10 | 23.93 ± 3.99% | 33.06 ± 3.69% | 39.61 ± 1.02% | 46.60 ± 3.11% | **49.29 ± 3.35%** |
| 7 | 0 | 3 | 22.13 ± 1.05% | 23.23 ± 1.94% | 20.80 ± 4.97% | 28.67 ± 1.51% | **32.53 ± 4.90%** |
| 5 | 3 | 2 | 25.13 ± 1.49% | 25.10 ± 1.72% | 28.73 ± 0.56% | **33.60 ± 6.24%** | 32.13 ± 3.57% |
| 5 | 0 | 5 | 16.53 ± 4.71% | 22.17 ± 3.68% | 28.13 ± 4.59% | 35.13 ± 3.36% | **36.80 ± 5.27%** |
| 4 | 3 | 3 | 27.60 ± 3.25% | 23.53 ± 8.40% | 25.87 ± 1.26% | 36.80 ± 4.67% | **43.53 ± 3.32%** |
| 3 | 0 | 7 | 23.53 ± 1.98% | 25.60 ± 5.01% | 31.77 ± 3.52% | 42.13 ± 2.19% | **44.93 ± 5.35%** |
| 0 | 7 | 3 | 24.93 ± 2.44% | 26.50 ± 4.06% | 30.33 ± 1.31% | 43.27 ± 3.27% | **43.73 ± 3.88%** |
| 0 | 5 | 5 | 24.70 ± 3.99% | 27.52 ± 3.69% | 32.06 ± 1.02% | 42.46 ± 3.11% | **44.42 ± 3.35%** |

# Results: Case Study

- Airsim drone navigation environment

- Airsim: autonomous navigation environment on quad-copter drones in a few indoor maps

- Experiment results

- Our model (SRTD, SRTD+ID) shows 5.01%~11.37% improvement over other baselines (SoftMod, PCGrad) in offline RL settings

Indoor-pyramid Indoor-complex

Indoor-gt Indoor-cloud

| Datasets | | | Comparison | | | Our model | |
|---|---|---|---|---|---|---|---|
| MR | RP | ME | TD3+BC | PCGrad | SoftMod | SRTD | SRTD+ID |
| 6 | 0 | 0 | $12.71 \pm 2.27\%$ | $15.70 \pm 0.34\%$ | $16.76 \pm 3.58\%$ | $22.34 \pm 0.98\%$ | $\mathbf{24.60 \pm 2.25\%}$ |
| 0 | 6 | 0 | $13.06 \pm 2.93\%$ | $13.45 \pm 0.70\%$ | $21.19 \pm 1.98\%$ | $29.34 \pm 0.32\%$ | $\mathbf{30.77 \pm 2.16\%}$ |
| 0 | 0 | 6 | $14.82 \pm 1.99\%$ | $16.93 \pm 2.15\%$ | $26.35 \pm 2.35\%$ | $30.36 \pm 2.61\%$ | $\mathbf{35.83 \pm 0.80\%}$ |
| 2 | 2 | 2 | $14.66 \pm 2.55\%$ | $16.38 \pm 4.63\%$ | $24.07 \pm 2.28\%$ | $\mathbf{29.08 \pm 2.36\%}$ | $28.37 \pm 1.09\%$ |
| 1 | 2 | 3 | $13.23 \pm 0.47\%$ | $14.12 \pm 3.09\%$ | $22.80 \pm 1.37\%$ | $27.78 \pm 2.21\%$ | $\mathbf{34.18 \pm 1.16\%}$ |
| 3 | 2 | 1 | $12.18 \pm 2.14\%$ | $12.28 \pm 1.92\%$ | $18.67 \pm 2.89\%$ | $25.47 \pm 2.61\%$ | $\mathbf{27.51 \pm 1.79\%}$ |

How to solve complex sequential decision problems
in the “pretrained” multi-task, RL context

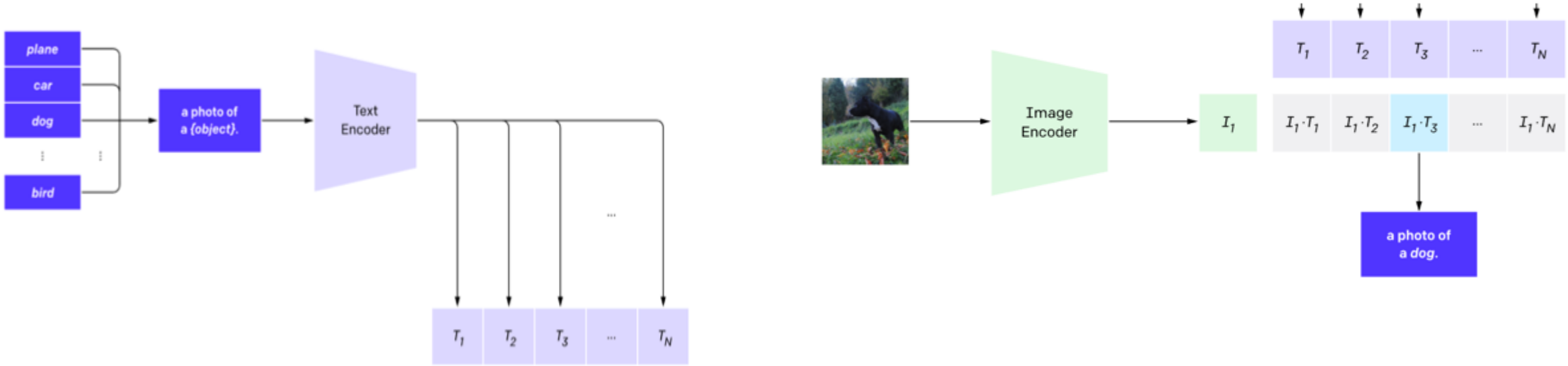
→

(1) How to use “pretrained” models
to solve complex sequential decision problems
in the multi-task, RL context

(2) How to learn a pretrained model efficiently

# Vision-language Zero-shot classification

- Object classification with zero-shot using vision-language models
  - Use the names of all the classes in the dataset as the set of potential text pairings and predict the most probable (image, text) pair according to CLIP



# Zero-shot object-goal navigation using Clip

- Use an image-goal navigation agent trained on Cl
- Object-goal navigation (language-goal navigation
  - The task of asking a virtual robot (agent) to find any insta
- Zero-shot object navigation
  - Train on the image-goal navigation : an agent finds the lo goal navigation, training “Semantic-goal navigation” – g

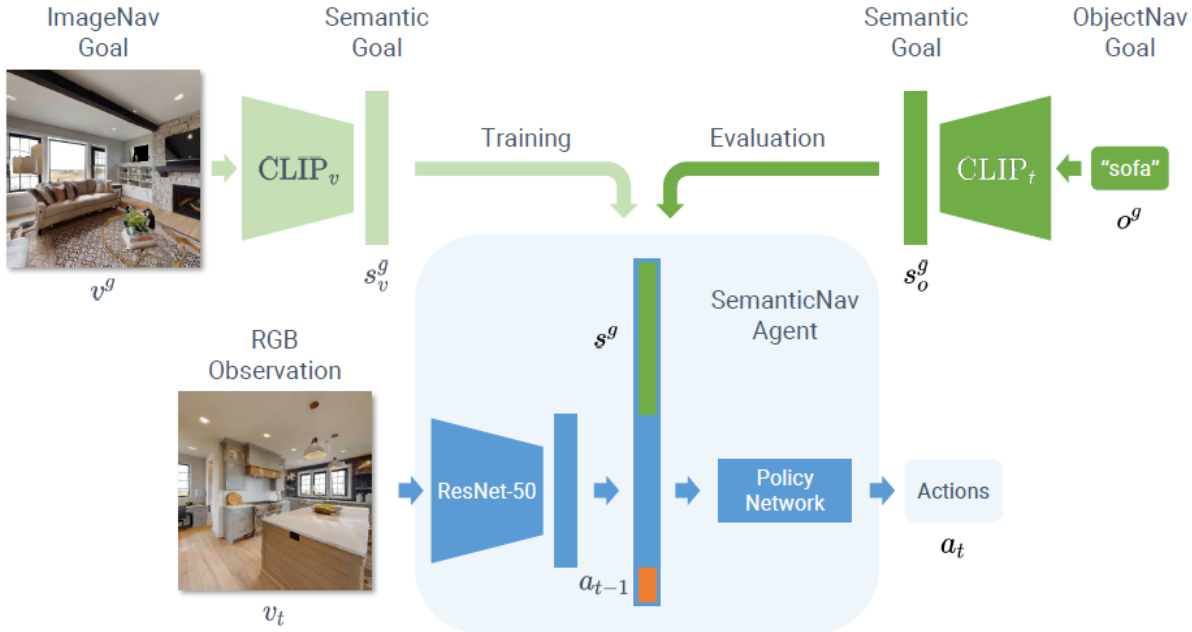



Figure 2: We tackle both `ImageNav` and `ObjectNav` via a common `SemanticNav` agent. This agent accepts a semantic goal embedding ($s^g$), which comes from either CLIP's visual encoder ($\texttt{CLIP}_v$) in ImageNav or CLIP's textual encoder ($\texttt{CLIP}_t$) in ObjectNav. Our agent has a simple architecture: RGB observations are encoded with a pretrained ResNet-50, and a recurrent policy network predicts actions using encodings of the goal $s^g$, observation, and the previous action $a_{t-1}$.

ZSON: Zero-Shot Object-Goal Navigation using Multimodal Goal Embeddings (Georgia Tech, 2022)

# Federated pretrained model in RL

- Limitations : sample inefficiency in RL and limited access to environments
- Federated offline RL

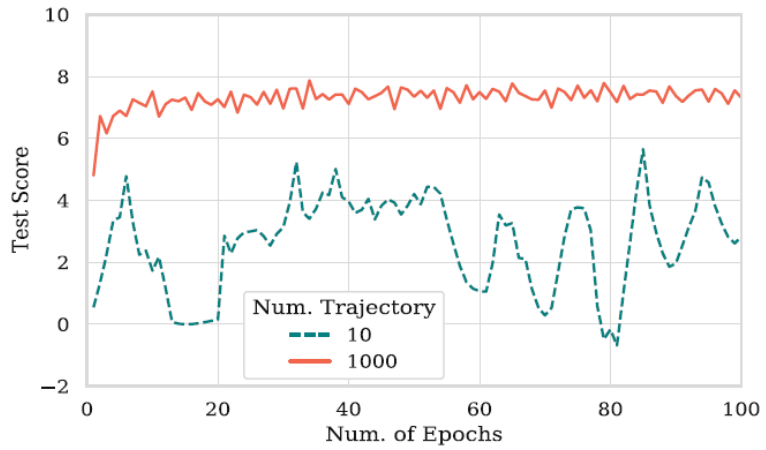



**Fig. 1.** The performance (test score) by offline RL (CQL) on the datasets of different trajectory sizes.

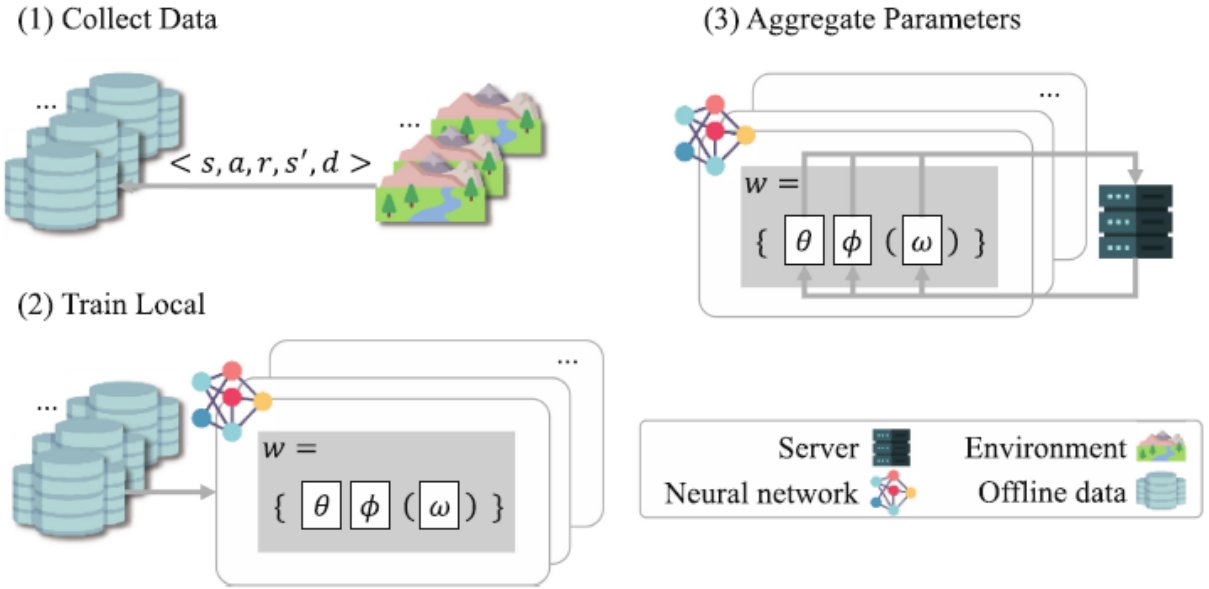



**Fig. 2.** Overall system concept. Datasets are collected from an environment. Each client locally trains its policy on the local dataset via the offline RL algorithm. The policy parameters are uploaded to the server after offline learning and aggregated by the federation algorithm. The aggregated parameters are then downloaded to clients, and the training procedure is repeated until convergence.

# Summary

- Pretrained models have gained great success in NLP and vision tasks
- Techniques for pretrained models in the RL context
  - Offline datasets and offline learning
  - Skill-based task-agnostic learning
- How to leverage pretrained models for RL
  - Multimodal pretrained models
- How to learn an RL pretrained model efficiently