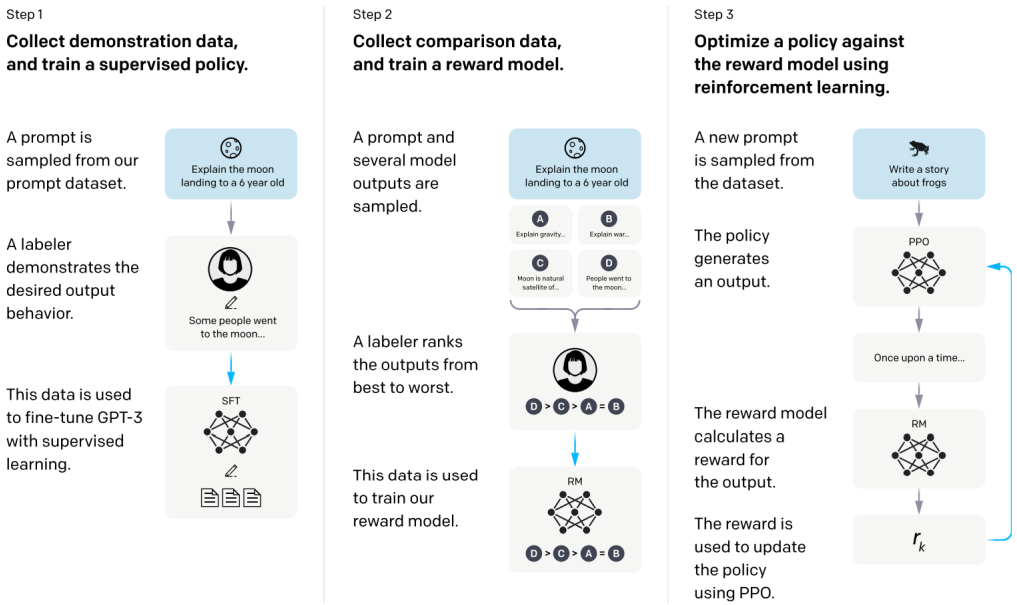
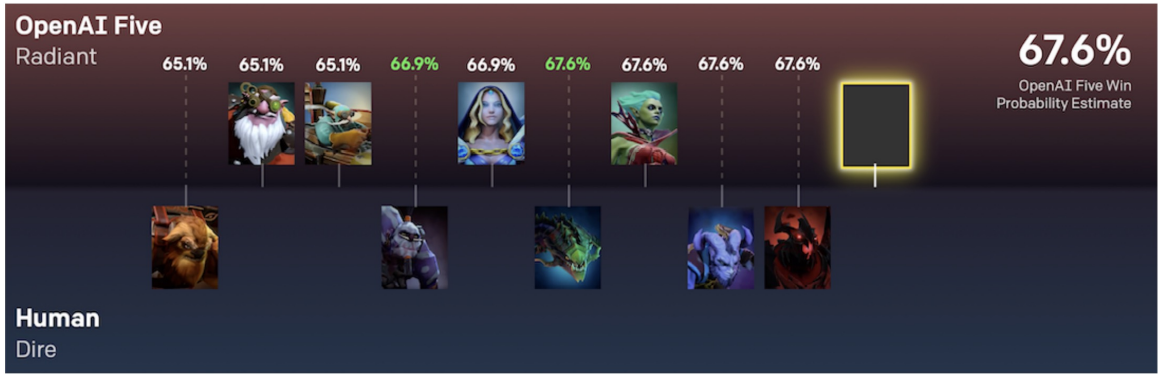
Visual Reinforcement Learning: A survey

Zhan Ling

Reinforcement learning(RL)



Training Language models



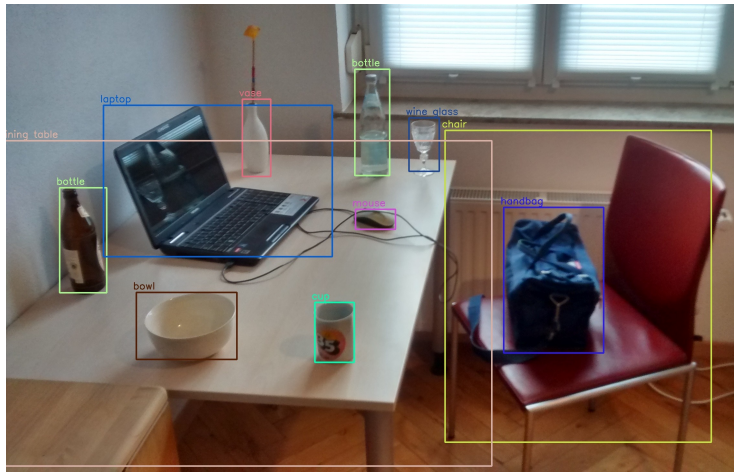
Multi-agent video game

AlphaGo

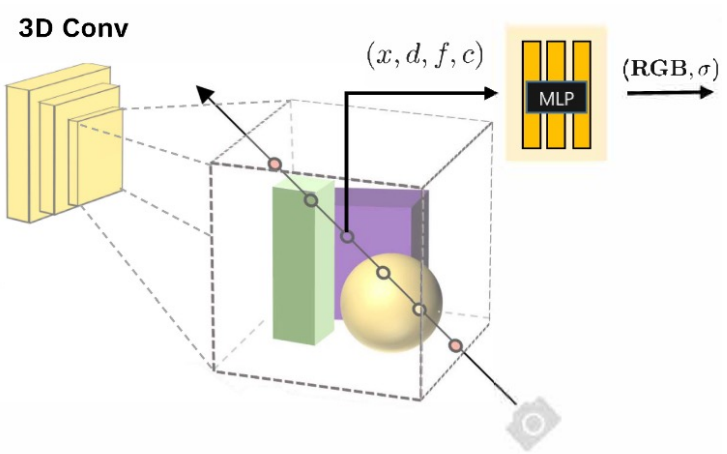
Single-agent chess game

Figures are from InstructGPT, OpenAI Five, Alpha Go

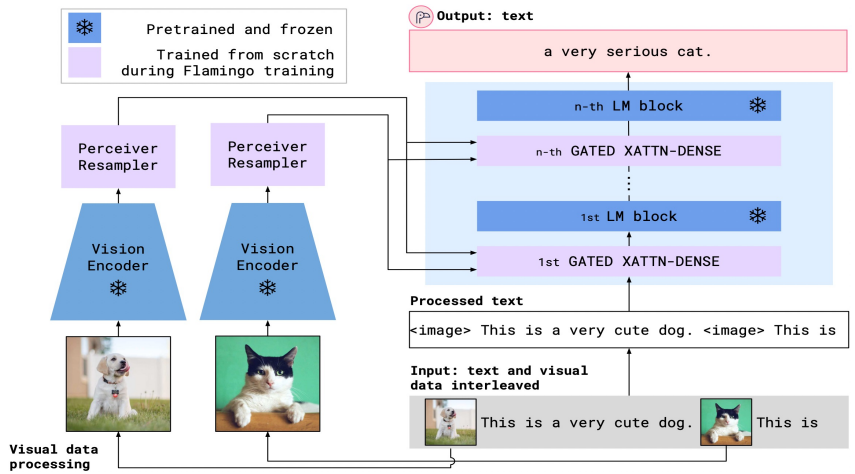
Computer vision



Object detection



3D reconstruction



Visual language model

Figures are from COCO, MVS-NeRF, Flamingo

Visual Reinforcement Learning(visual RL) - A challenging direction in deep RL

Autonomous driving

Generalizable robot manipulation

Video games

Figures are from CARLA, ManiSkill, Procgen

Outline

- Reinforcement Learning
- Visual Reinforcement Learning (Visual RL)
  - Model-free algorithms
  - Model-based algorithms
  - Parallel training frameworks
- Recent Trends in Visual RL
  - Representation learning for visual RL
  - Visual RL in the real world
  - Future directions

Reinforcement Learning

Basic concepts

Reinforcement Learning(RL)

- At each step t
  - The agent determines the action A_t.
  - The environment execute the action A_t.
  - The agent receive the new observation O_{t+1} and reward r_t from the environment.
- Aim: Find an agent that achieves maximum accumulated rewards from the environment.

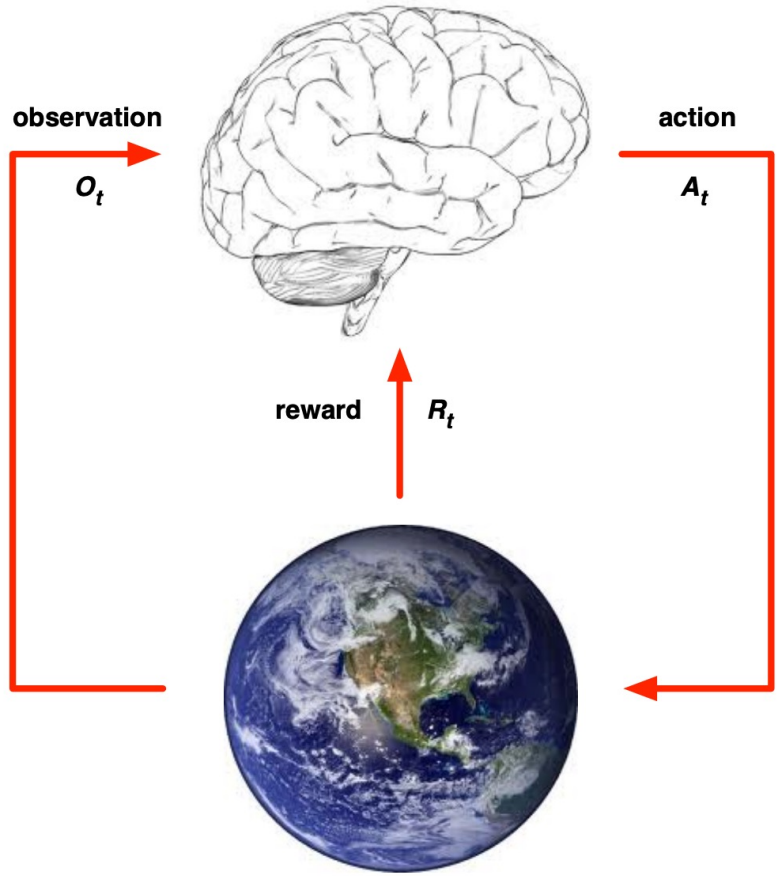


Image source https://www.davidsilver.uk/wp-content/uploads/2020/03/intro_RL.pdf

Problem formulation

- RL problems consist of an environment, an agent, a learner.
- Environment is an MDP: $M = (S, A, T, R, \rho_0, \gamma)$.
  - S, A are the state and action space respectively.
  - $T(s'|s,a)$: $S \times A \times S \to \mathbb{R}$ is the transition function.
  - $r(s,a)$: $S \times A \to \mathbb{R}$ is the reward function.
  - $\rho_0(s)$: $S \to \mathbb{R}$ is the initial state distribution.
  - $\gamma \in [0,1]$ is the discounted factor.
- Agent is a function maps from state to action: $\pi(a|s)$: $S \times \mathrm{A} \to \mathbb{R}$.
- Leaner optimizes the agent π by maximizing:

$$J(\pi) = E_{s_0, a_0 \sim \pi(a|s_0), \dots}[\sum_{t=0} \gamma^t \, r(s_t, a_t)\,]$$

Visual Reinforcement Learning

Algorithms and systems

Two types of visual RL algorithms

- Model-free visual RL = Deep model-free RL + vision networks
  - Use vision networks in the agent to support visual observations.

- Model-based visual RL
  - Learn a world-model from visual observations.
  - Use model-free visual RL to learn the agent.

Model-free algorithms

- Two classes of algorithms:
  - Value-based algorithms, e.g., DQN, SAC
  - Policy-based algorithms, e.g., PPO

Algorithm 1 Pipleine of model-free RL

Initialize policy π and replay buffer $\mathbf{D}$.
while not converged **do**
  ▷ collect experiences from the real environment.
  $\mathbf{D} \leftarrow \mathbf{D} \cup \text{COLLECT}(\mathbf{env}, \pi)$
  ▷ update policy using data from **env**.
  $\pi \leftarrow \text{OPTIMIZE\_POLICY}(\pi, \mathbf{D})$
end while

Value-based methods

- A state-value function Q is learned through Bellman equation.

$$Q(s_0, a_0) = \mathbb{E}_{s_1, a_1, \cdots}\left[\sum_{t=0} \gamma^t r(s_t, a_t)\right]$$

$$\begin{aligned} Q_T(s,a) &= r(s,a) + \gamma Q^{w'}(s', \pi^\theta(s')) \\ L_{TD} &= \mathbb{E}_{s,a}\left[\|Q^w(s,a) - Q_T(s,a)\|\right] \\ w &\leftarrow w - \alpha \nabla_w L_{TD} \end{aligned}$$

- The policy is optimized based on the learned Q function:
  - Discrete action space: $\pi(s) = \arg\max_a Q^w(s,a)$
  - Continuous action space: $\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_s\left[Q^w(s, \pi^\theta(s))\right]$

Policy-based methods - REINFORCE

- Expected discounted accumulated reward:

$$J(\theta) = \mathbb{E}_{\tau}\left[r_{\gamma}(\tau)\right] = \mathbb{E}_{s_0, a_0, \cdots}\left[\sum_{t=0} \gamma^t r(s_t, a_t)\right]$$

- Policy gradient & REINFORCE algorithm:

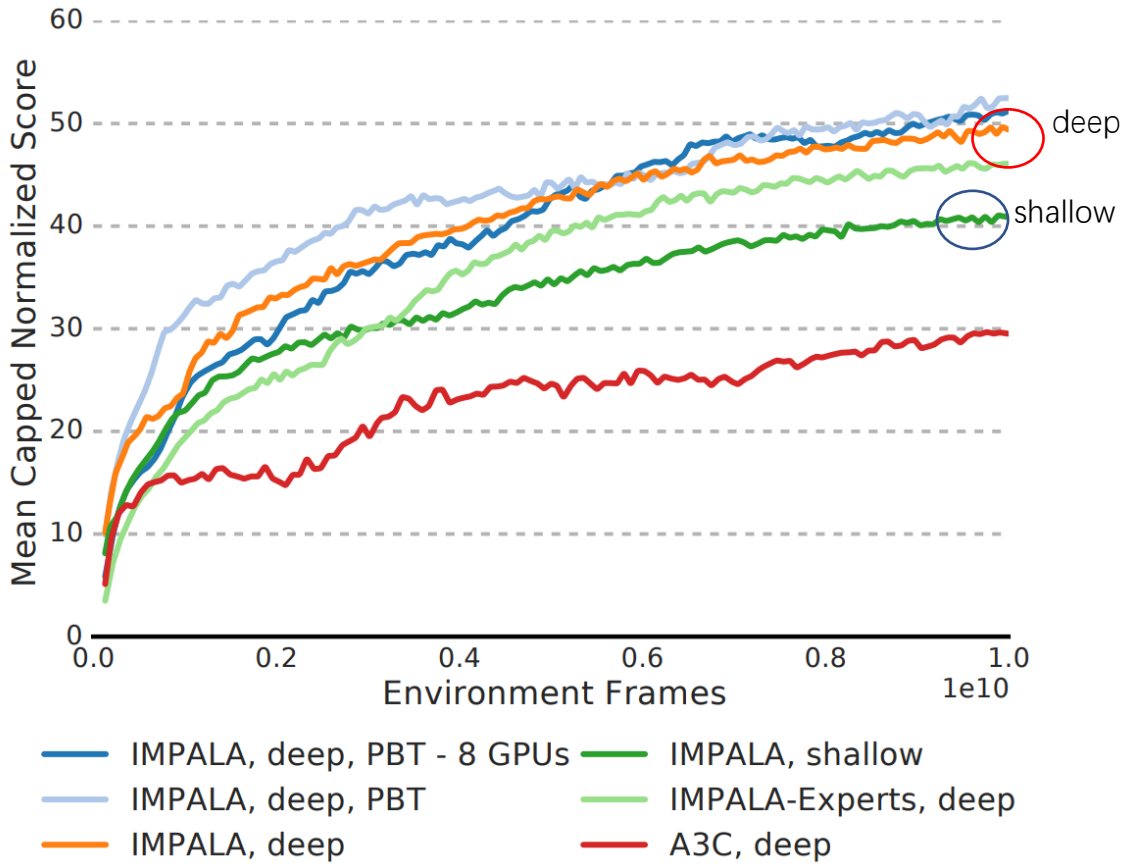
$$\begin{aligned}
\nabla_{\theta} J(\theta) =& \nabla_{\theta} \mathbb{E}_{\tau}\left[r_{\gamma}(\tau)\right] \\
=& \mathbb{E}_{\tau}\left[\nabla_{\theta} \log p^{\theta}(\tau) r_{\gamma}(\tau)\right] \\
=& \mathbb{E}_{\tau}\left[\left(\sum_{t} \nabla_{\theta} \log \pi^{\theta}(a_t|s_t)\right) r_{\gamma}(\tau)\right]
\end{aligned}$$

Policy-based methods – Actor-critic

- Estimation of policy gradient from sampled trajectories has high variance.
- Policy gradient theorem $\nabla_\theta J(\theta) = \mathbb{E}_s \left[\nabla_\theta \log \pi^\theta(a|s) Q(s,a) \right]$
- Actor-critic estimates the policy gradient with a learned action-value function $Q(s,a)$ and reduce the variance.
- Using advantage function $A(s,a) = Q(s,a) - V(s)$ instead of $Q(s,a)$ can further reduce the variance.

Vision networks for model-free visual RL

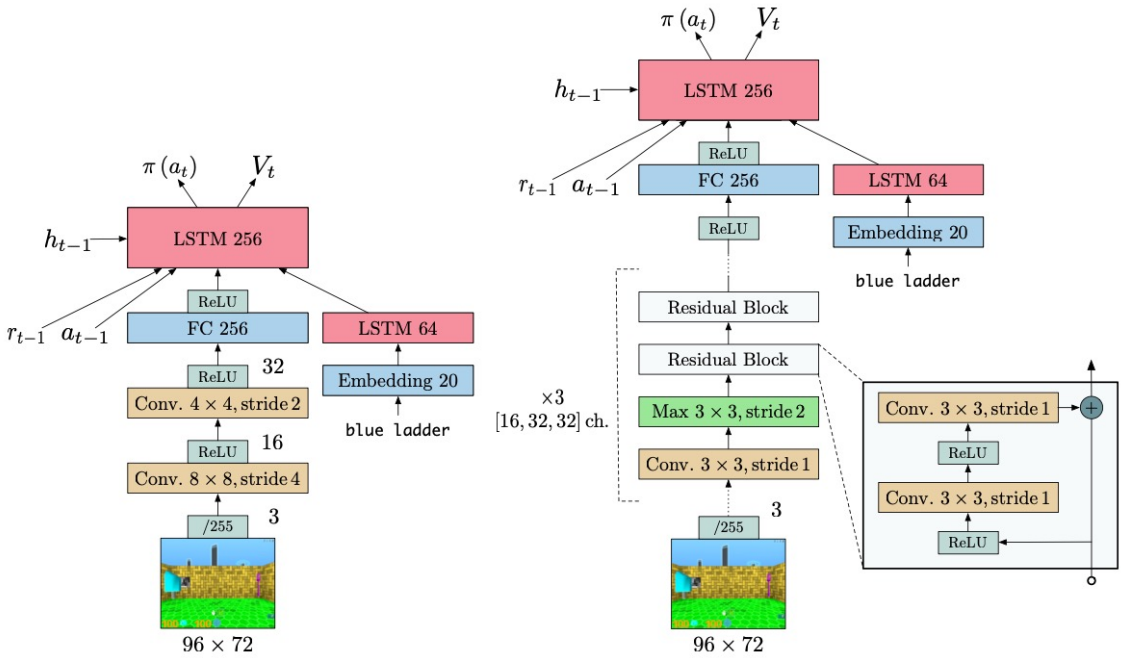
- 3-layer or 4-layer CNNs in most visual RL algorithms.
- Deeper networks may learn better agents!



Figures are from IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures

Partial observations in visual RL

- Challenge: partial observations, e.g., images
- Solution: recurrent networks, e.g., LSTM, GRU



Figures are from IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures

Model-based algorithms

- Two directions of learning a world-model
  - With visual reconstruction, e.g., PlaNet, Dreamer
  - Without visual reconstruction, e.g., MuZero, EfficientZero, TD-MPC

Algorithm 2 Pipleine of model-based RL

Initialize policy π, world model $\mathbf{T}$, and replay buffer $\mathbf{D}$.
while not converged **do**
  ▷ collect experiences from the real environment.
  $\mathbf{D} \leftarrow \mathbf{D} \cup \text{COLLECT}(\mathbf{env}, \pi)$
  ▷ update the model θ using collected data.
  $\mathbf{T} \leftarrow \text{TRAIN\_MODEL}(\mathbf{T}, \mathbf{D})$
  ▷ update policy using the model $\mathbf{T}$ and collected data.
  $\pi \leftarrow \text{OPTIMIZE\_POLICY}(\pi, \mathbf{T}, \mathbf{D})$
end while

Model-based algorithms

- With visual reconstruction
  - PlaNet
  - Dreamer
  - Dreamer-v2

PlaNet

- World model: recurrent state-space model(RSSM)

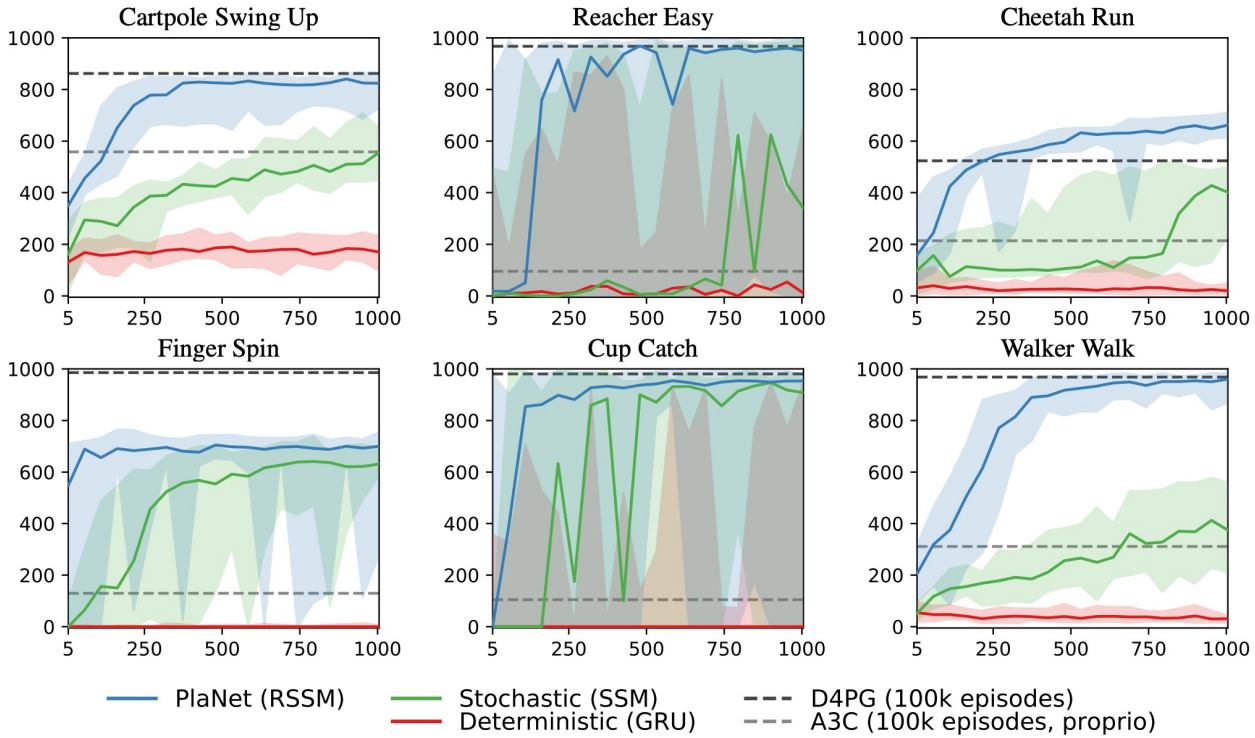
$$\begin{aligned}
&\textbf{Deterministic state model:} && h_t = f(h_{t-1}, s_{t-1}, a_{t-1}) \\
&\textbf{Stochastic state model:} && s_t \sim p(s_t \mid h_t) \\
&\textbf{Observation model:} && o_t \sim p(o_t \mid h_t, s_t) \\
&\textbf{Reward model:} && r_t \sim p(r_t \mid h_t, s_t),
\end{aligned}$$

- Policy = RSSM + Model Predictive Control(MPC)
  - Sample a set of trajectories of actions and execute them in RSSM.
  - Use CEM to find the best trajectory based on rewards from RSSM.

Figures are from Learning Latent Dynamics for Planning from Pixels

Results of PlaNet

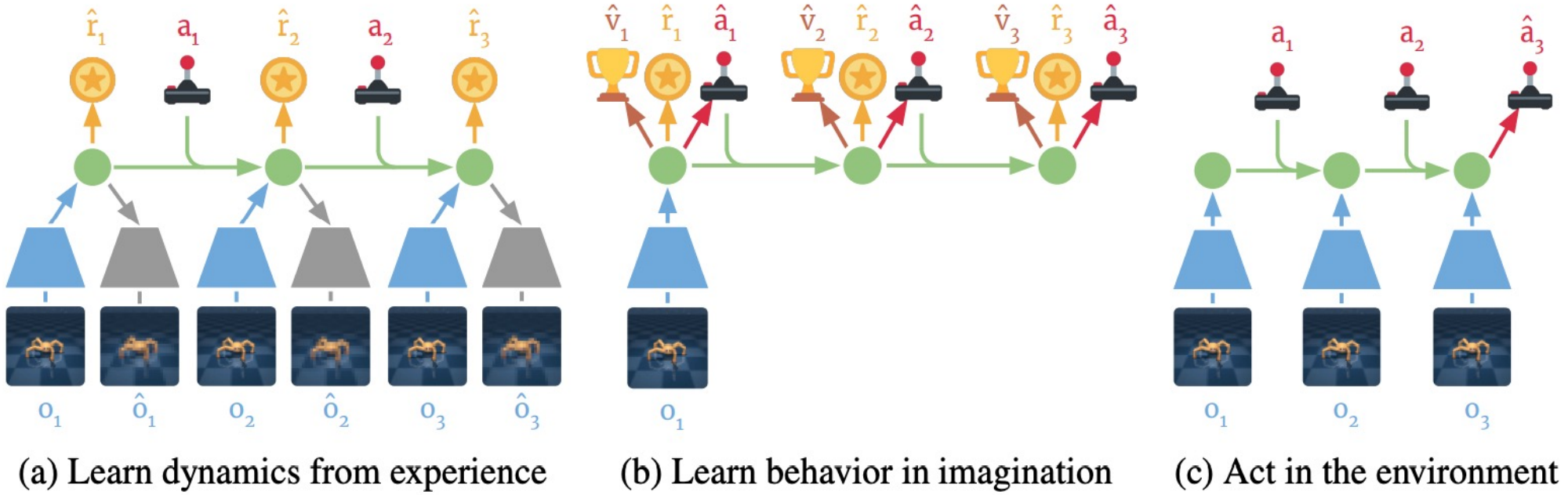
- Better performance with 200x fewer samples than best model-free methods.



Figures are from Learning Latent Dynamics for Planning from Pixels

Dreamer

- Dreamer: PlaNet & model-free visual RL

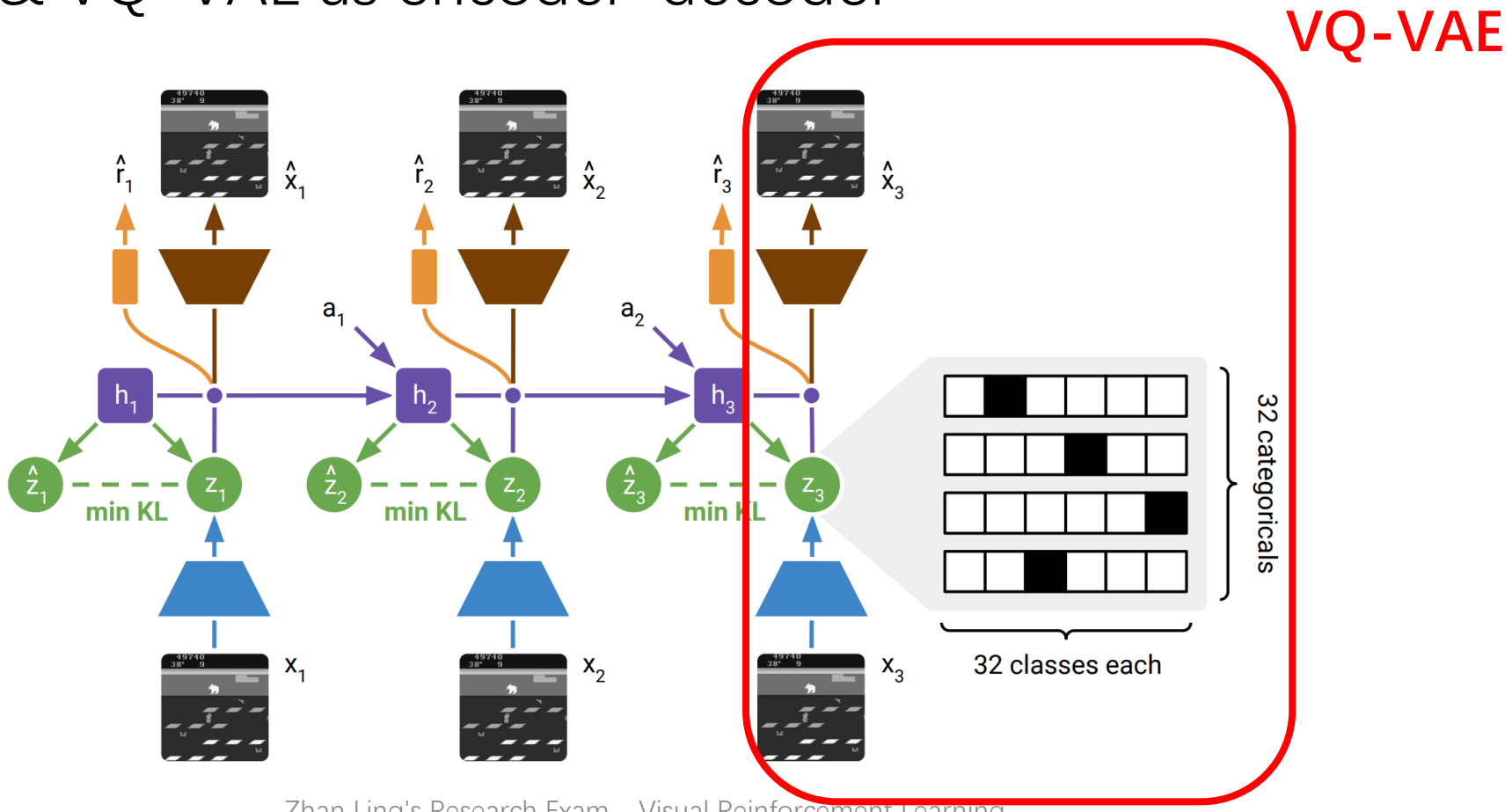


(a) Learn dynamics from experience (b) Learn behavior in imagination (c) Act in the environment

Figures are from Dream to Control: Learning Behaviors by Latent Imagination

Dreamer-v2

- Dreamer & VQ-VAE as encoder-decoder



Figures are from Mastering Atari with Discrete World Models

Model-based algorithms

- Without visual reconstruction
  - MuZero
  - EfficientZero
  - TD-MPC

MuZero

- World model consists of:
  - Dynamics: $\mathbf{s_t} \sim f(s_t|s_{t-1}, a_{t-1})$
  - Reward: $r_t \sim p(r_t|s_{t-1}, a_{t-1})$
- Plan using MCTS with learned world model.
- **Without reconstructing the observation!**
  - Faster
  - Easier to support irregular inputs
    - e.g., pointclouds

MuZero on Atari Games

Agent	Median	Mean	Env. Frames	Training Time	Training Steps
Ape-X [18]	434.1%	1695.6%	22.8B	5 days	8.64M
R2D2 [21]	1920.6%	4024.9%	37.5B	5 days	2.16M
MuZero	**2041.1%**	**4999.2%**	20.0B	12 hours	1M
IMPALA [9]	191.8%	957.6%	200M	–	–
Rainbow [17]	231.1%	–	200M	10 days	–
UNREAL[a] [19]	250%[a]	880%[a]	250M	–	–
LASER [36]	431%	–	200M	–	–
MuZero Reanalyze	**731.1%**	**2168.9%**	200M	12 hours	1M

Table 1: **Comparison of *MuZero* against previous agents in Atari**. We compare separately against agents trained in large (top) and small (bottom) data settings; all agents other than *MuZero* used model-free RL techniques. Mean and median scores are given, compared to human testers. The best results are highlighted in **bold**. *MuZero* sets a new state of the art in both settings. [a]Hyper-parameters were tuned per game.

Figures are from Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model

EfficientZero

- Self-supervised consistency loss
  - learn the dynamics in the latent space
- Predict reward prefix instead of per-step reward.

$$\sum_{i=0}^{k-1} \gamma^i r_{t+i}$$

- Correct the target value with MCTS over learned model.

$$z_t = \sum_{i=0}^{l-1} \gamma^i u_{t+i} + \gamma^l \nu_{t+l}^{\text{MCTS}}$$

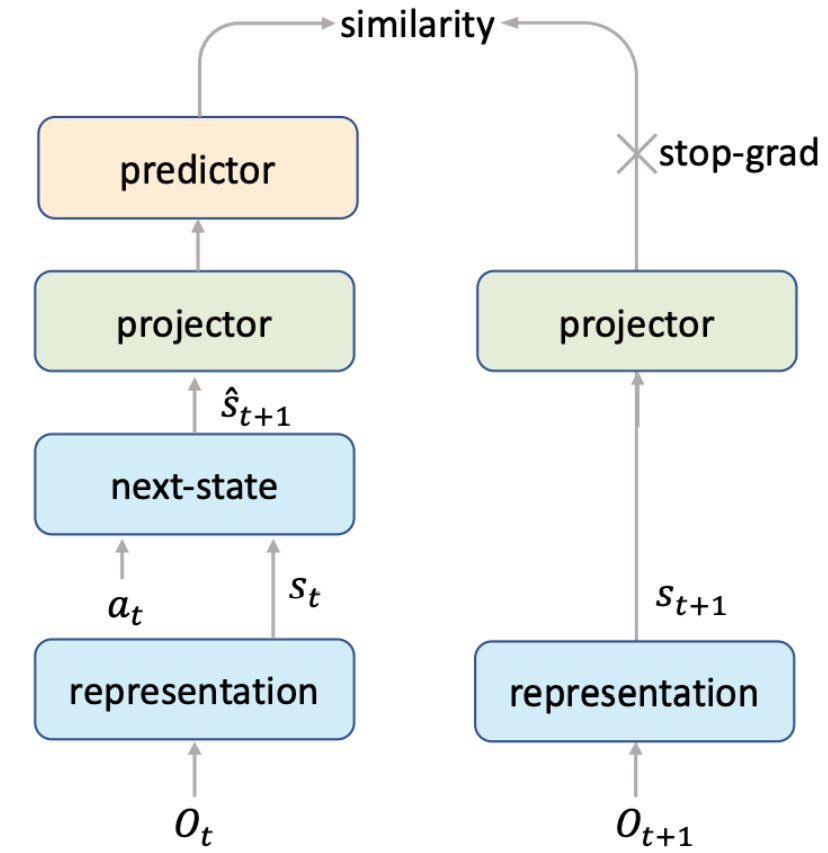


Figure 2: The self-supervised consistency loss.

Figures are from Mastering Atari Games with Limited Data

TD-MPC

- Continuous version of MuZero
  - Replace MCTS with MPC

	Model-free				Model-based				Ours
100k env. steps	SAC State	SAC Pixels	CURL	DrQ	PlaNet	Dreamer	MuZero*	Eff.Zero*	**TD-MPC**
Cartpole Swingup	812±45	419±40	597±170	**759±92**	563±73	326±27	219±122	**813±19**	**770±70**
Reacher Easy	919±123	145±30	517±113	601±213	82±174	314±155	493±145	**952±34**	628±105
Cup Catch	957±26	312±63	772±241	**913±53**	710±217	246±174	542±270	**942±17**	**933±24**
Finger Spin	672±76	166±128	779±108	**901±104**	560±77	341±70	–	–	**943±59**
Walker Walk	604±317	42±12	344±132	**612±164**	221±43	277±12	–	–	**577±208**
Cheetah Run	228±95	103±38	**307±48**	**344±67**	165±123	235±137	–	–	222±88

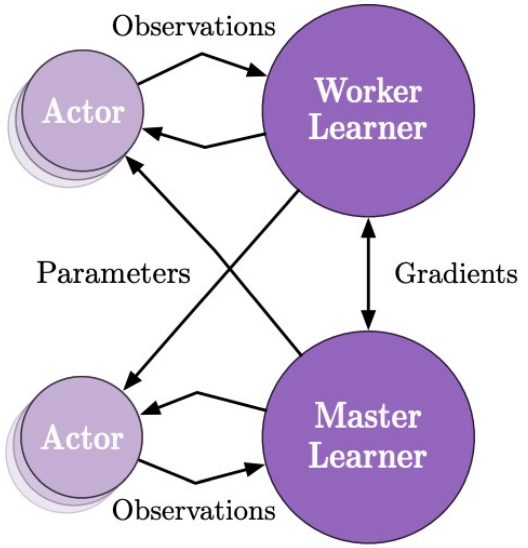
Figures are from Mastering Atari Games with Limited Data

Parallel training frameworks

- Motivation:
  - Training a visual RL agent may need millions of samples.
  - Environments are usually slow: rendering + simulation.
  - Visual networks even slows down the training speed.
- Systems:
  - IMPALA
  - SEED
  - Envpool
  - DD-PPO

IMPALA

- Decouple agent training(**Master Learner**) from sample collection(**Actor**).
- Use the same latest policy from learner to collect the new trajectory.
- Use V-trace (off-policy) in leaner.



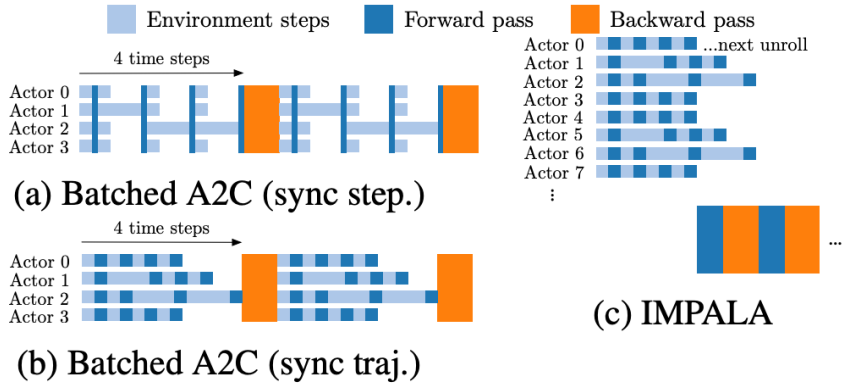


Figure 2. Timeline for one unroll with 4 steps using different architectures. Strategies shown in **(a)** and **(b)** can lead to low GPU utilisation due to rendering time variance within a batch. In **(a)**, the actors are synchronised after every step. In **(b)** after every n steps. IMPALA **(c)** decouples acting from learning.

Figures are from IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures

Speed of IMPALA

- Around 20K FPS with a 48-core CPU

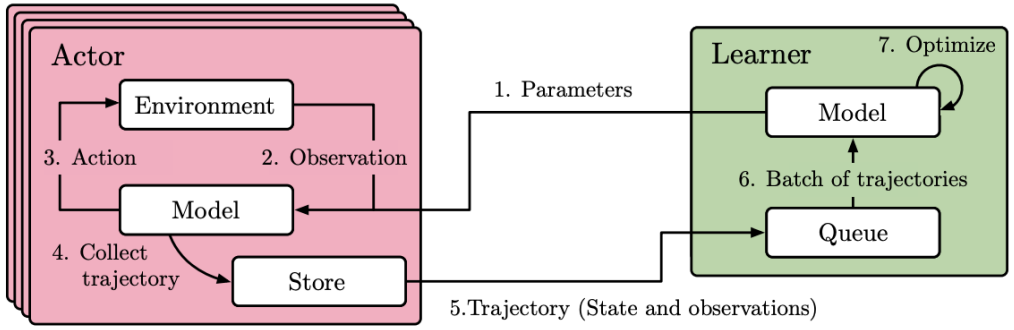
Architecture	CPUs	GPUs[1]	FPS[2]	
Single-Machine			Task 1	Task 2
A3C 32 workers	64	0	6.5K	9K
Batched A2C (sync step)	48	0	9K	5K
Batched A2C (sync step)	48	1	13K	5.5K
Batched A2C (sync traj.)	48	0	16K	17.5K
Batched A2C (dyn. batch)	48	1	16K	13K
IMPALA 48 actors	48	0	17K	20.5K
IMPALA (dyn. batch) 48 actors[3]	48	1	21K	24K
Distributed				
A3C	200	0	46K	50K
IMPALA	150	1	80K	
IMPALA (optimised)	375	1	200K	
IMPALA (optimised) batch 128	500	1	250K	

[1] Nvidia P100 [2] In frames/sec (4 times the agent steps due to action repeat). [3] Limited by amount of rendering possible on a single machine.

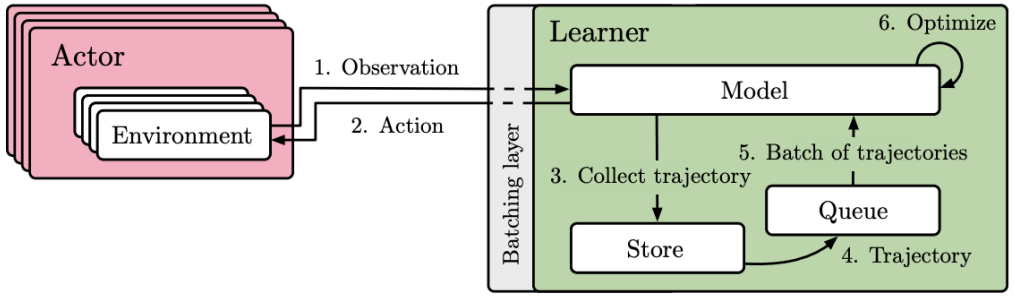
Figures are from IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures

SEED

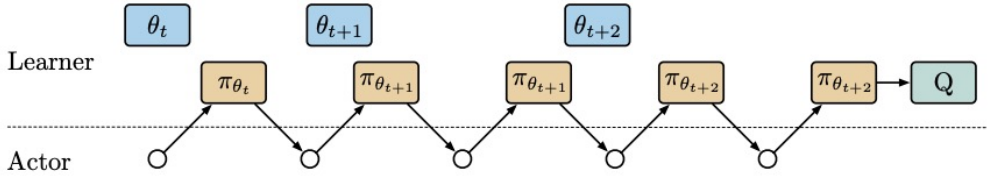
- Move the network inference part from actor to learner.
- Shares the same policy network for all actors to reduce the off-policy effect.



(a) IMPALA architecture (distributed version)



(b) SEED architecture, see detailed replay architecture in Figure 3.



(b) Off-policy in SEED. Optimizing a model has immediate effect on the policy. Thus, the trajectory consists of actions sampled from many different policies $(\pi_{\theta_t}, \pi_{\theta_{t+1}}, ...)$.

Figures are from SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference

Speed of SEED

- Around 74K FPS with a 104-core CPU

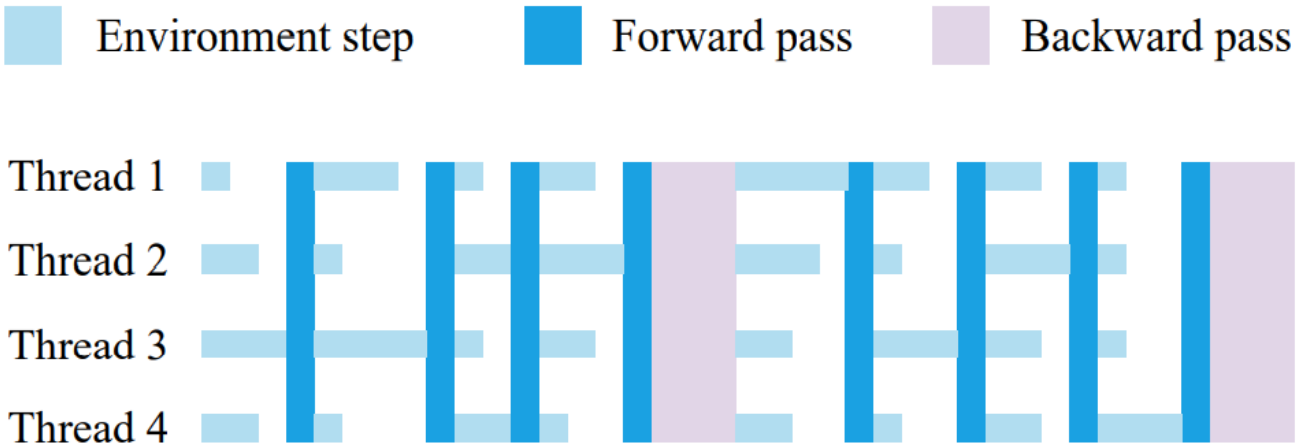
Architecture	Accelerators	Environments	Actor CPUs	Batch Size	FPS	Ratio
DeepMind Lab						
IMPALA	Nvidia P100	176	176	32	30K	—
SEED	Nvidia P100	176	44	32	19K	**0.63x**
SEED	TPU v3, 2 cores	312	104	32	74K	**2.5x**
SEED	TPU v3, 8 cores	1560	520	48[1]	330K	**11.0x**
SEED	TPU v3, 64 cores	12,480	4,160	384[1]	2.4M	**80.0x**
Google Research Football						
IMPALA, Default	2 x Nvidia P100	400	400	128	11K	—
SEED, Default	TPU v3, 2 cores	624	416	128	18K	**1.6x**
SEED, Default	TPU v3, 8 cores	2,496	1,664	160[3]	71K	**6.5x**
SEED, Medium	TPU v3, 8 cores	1,550	1,032	160[3]	44K	—
SEED, Large	TPU v3, 8 cores	1,260	840	160[3]	29K	—
SEED, Large	TPU v3, 32 cores	5,040	3,360	640[3]	114K	**3.9x**
Arcade Learning Environment						
R2D2	Nvidia V100	256	N/A	64	85K[2]	—
SEED	Nvidia V100	256	55	64	67K	**0.79x**
SEED	TPU v3, 8 cores	610	213	64	260K	**3.1x**
SEED	TPU v3, 8 cores	1200	419	256	440K[4]	**5.2x**

[1] 6/8 cores used for training. [2] Each of the 256 R2D2 actors run at 335 FPS (information from the R2D2 authors). [3] 5/8 cores used for training. [4] No frame stacking.

Figures are from SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference

Envpool

- Asynchronous environment step to avoid waiting the slow environments in the same batch.
- Rewrite the environment in C++.



(a) Sync step, numenv = batch_size = 4

(b) Async step, numenv = 4, batch_size = 3

Figures are from EnvPool: A Highly Parallel Reinforcement Learning Environment Execution Engine

Envpool

- Very fast! Over 200K steps on a workstation

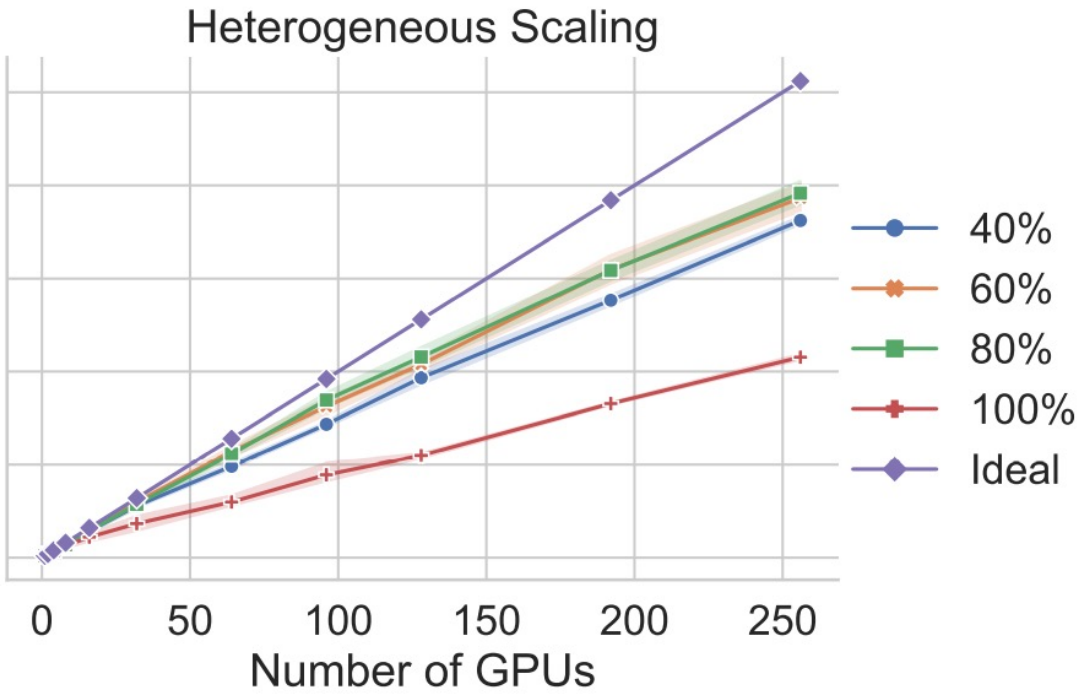
Table 1: Numeric results for benchmarking.

System Configuration	Laptop		Workstation		DGX-A100	
Method \ Env (FPS)	Atari	MuJoCo	Atari	MuJoCo	Atari	MuJoCo
For-loop	4,893	12,861	7,914	20,298	4,640	11,569
Subprocess	15,863	36,586	47,699	105,432	71,943	163,656
Sample-Factory	28,216	62,510	138,847	309,264	707,494	1,573,262
EnvPool (sync)	37,396	66,622	133,824	380,950	427,851	949,787
EnvPool (async)	**49,439**	**105,126**	**200,428**	**582,446**	891,286	2,363,864
EnvPool (numa+async)	/	/	/	/	**1,069,922**	**3,134,287**

Figures are from EnvPool: A Highly Parallel Reinforcement Learning Environment Execution Engine

DD-PPO

- For parallel training on multiple GPUs.
- On each GPU, a separate worker is used to collect samples.
- The time of collecting the same number of samples over the same GPU can vary a lot.
- DD-PPO enables near-identical scaling by stopping when 80% of workers finish the sample collection.



Figures are from DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames

Representation learning for Visual RL

Improving sample efficiency with better visual representation

Deep representation learning in visual RL

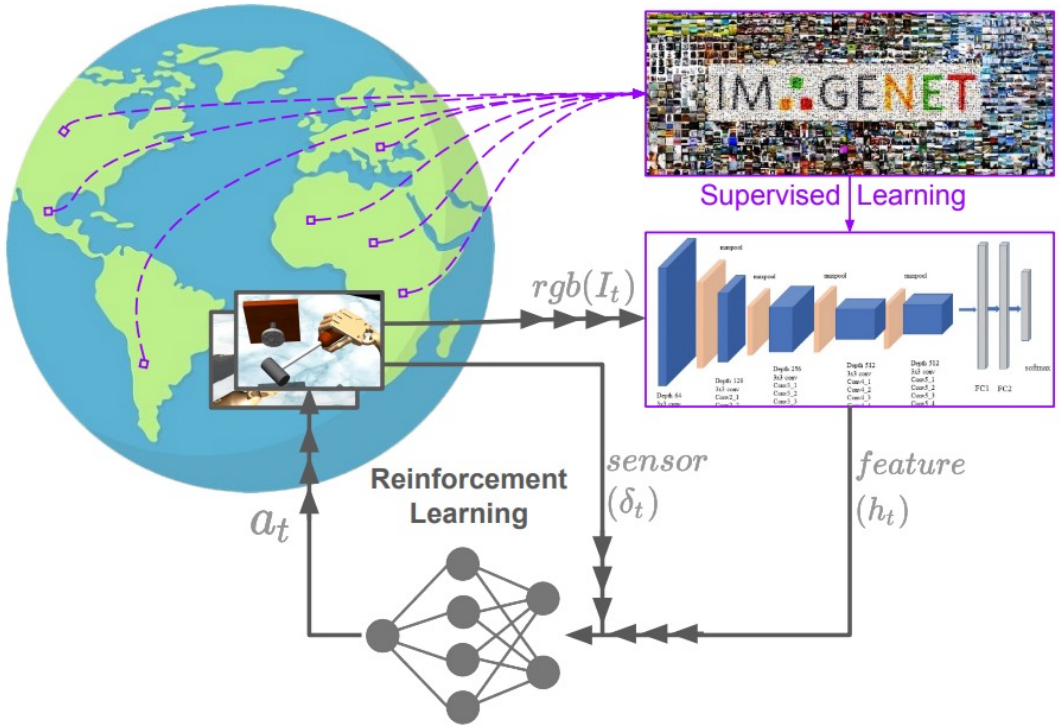
- Two directions:
  - Pre-training can provide a good initialization or a good representation.
  - Using self-supervised learning or data-augmentation to learn better representation during visual RL learning.

Pre-training for visual RL

- **Single-view pre-training**
  - RRL
  - PVR
  - MVP
- Multi-view pre-training
  - NeRF-RL
- Video pre-training
  - Transporter
  - R3M
  - APV

RRL

- Model, e.g., ResNet, pre-trained with supervised learning on ImageNet, provides representation for visual RL training.



Figures are from RRL: Resnet as representation for Reinforcement Learning

RRL results over manipulation tasks

- ImageNet features are effective for visual RL tasks, e.g., robot manipulation.
- Model types may influence the performance.

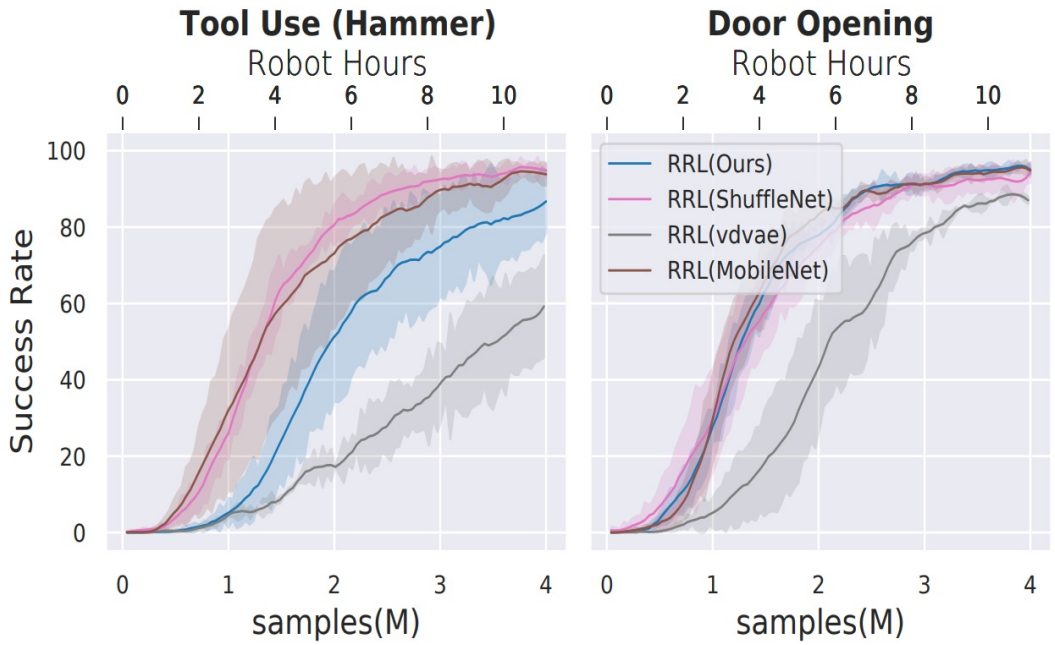


Figure 6. Effect of different types of Feature extractor pretrained on ImageNet dataset, highlighting that not just Resnet but any feature extractor pretrained on a sufficiently wide distribution of data remains effective.

Figures are from RRL: Resnet as representation for Reinforcement Learning

PVR

- Compare pre-training with supervised and unsupervised learning.
- All methods are effective and better than training from scratch.

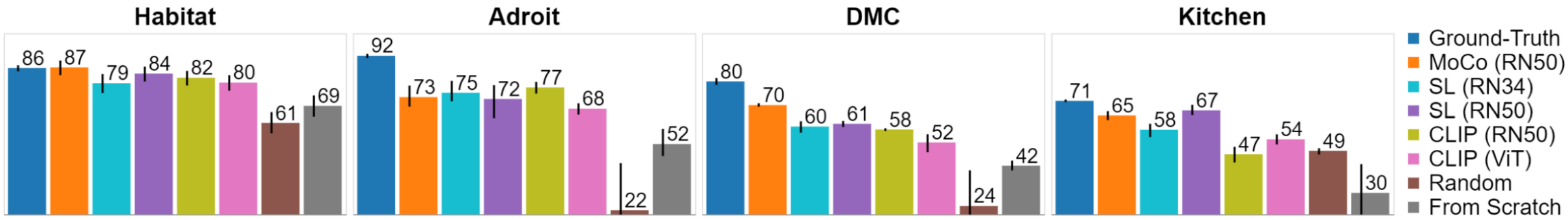


Figure 6: **Success rate of off-the-shelf PVRs**. Numbers at the top of the bar report mean values over five seeds, while thin black lines denote 95% confidence intervals. SL refers to standard supervised learning as in (He et al., 2016). Any PVR is better than training the perception end-to-end from scratch together with the control policy. In Habitat, MoCo matches the performance of ground-truth features. On the contrary, in MuJoCo, no off-the-shelf PVR can match ground-truth features.

Figures are from The (Un)Surprising Effectiveness of Pre-Trained Vision Models for Control

MVP

- Masked autoencoder(MAE) pre-training
- Dataset: HOI

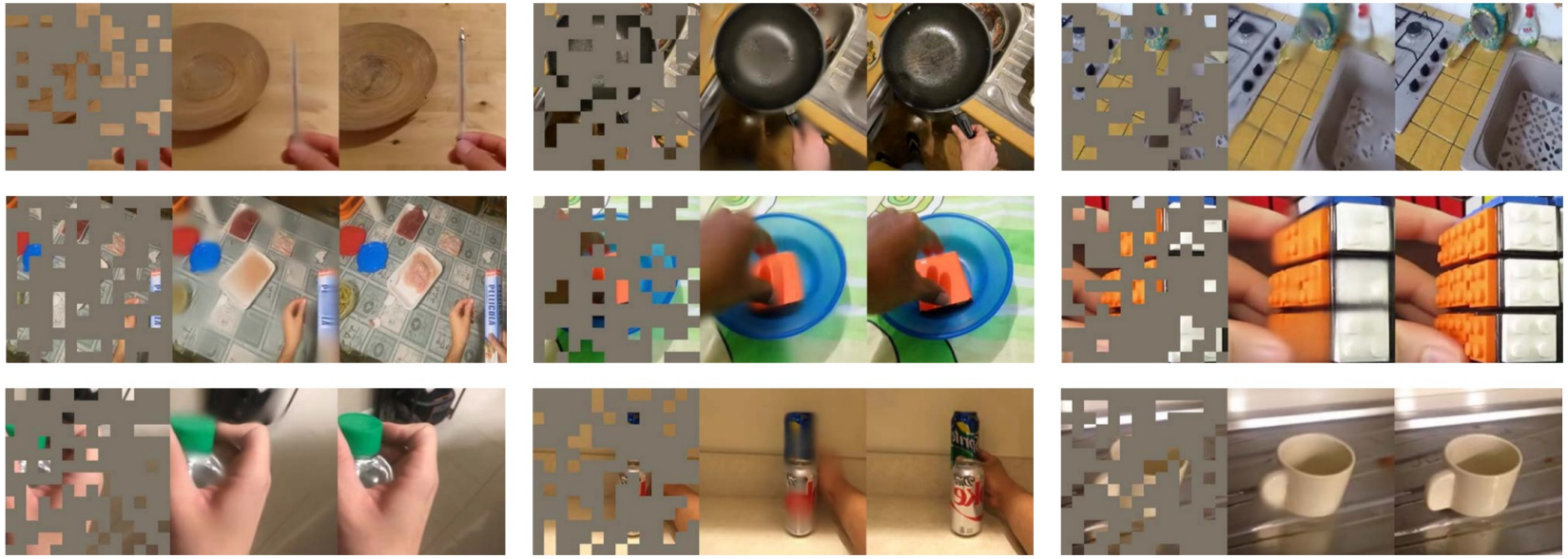
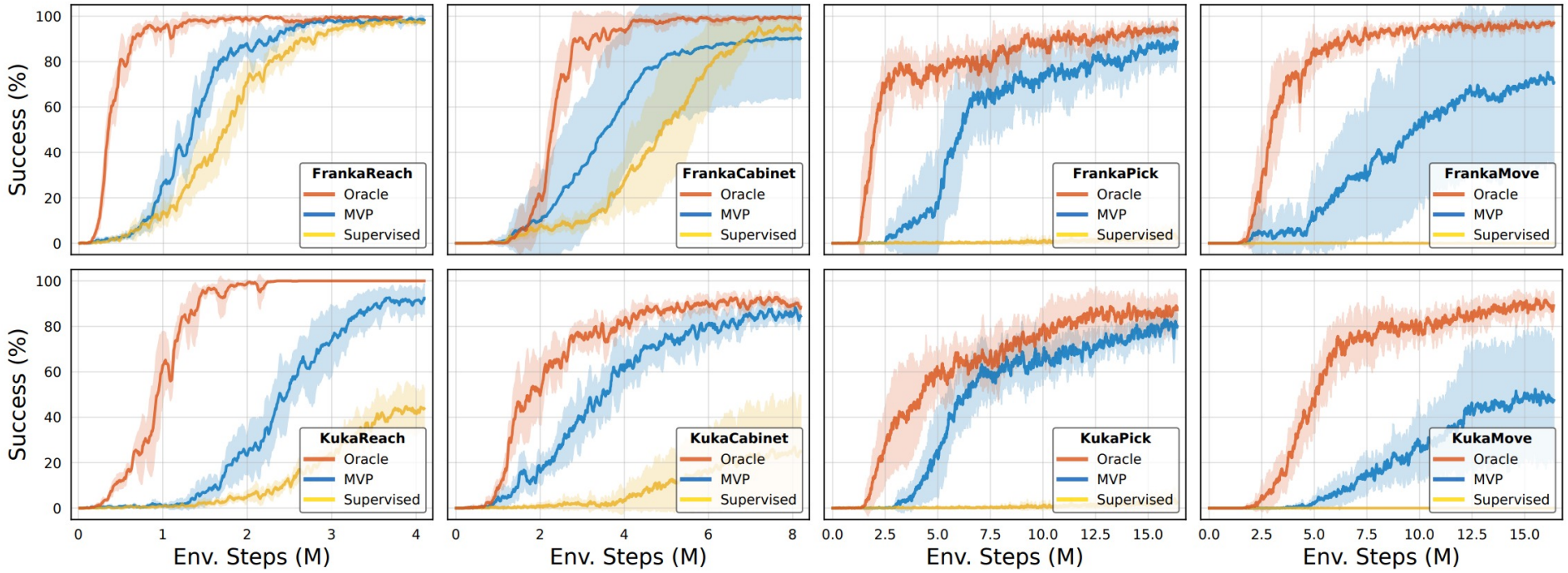


Figure 3. **Example reconstructions.** For each triplet from left to right: the masked image, the reconstructed image, the ground-truth target. We observe that the autoencoder learns color, shape, and object affordance using self-supervision from in-the-wild images.

Figures are from Masked Visual Pre-training for Motor Control

MVP

- MVP performs better than model pre-trained with supervised learning.



Figures are from Masked Visual Pre-training for Motor Control

Pre-training for visual RL

- Single-view pre-training
  - RRL
  - PVR
  - MVP
- **Multi-view pre-training**
  - NeRF-RL
- Video pre-training
  - Transporter
  - R3M
  - APV

NeRF-RL

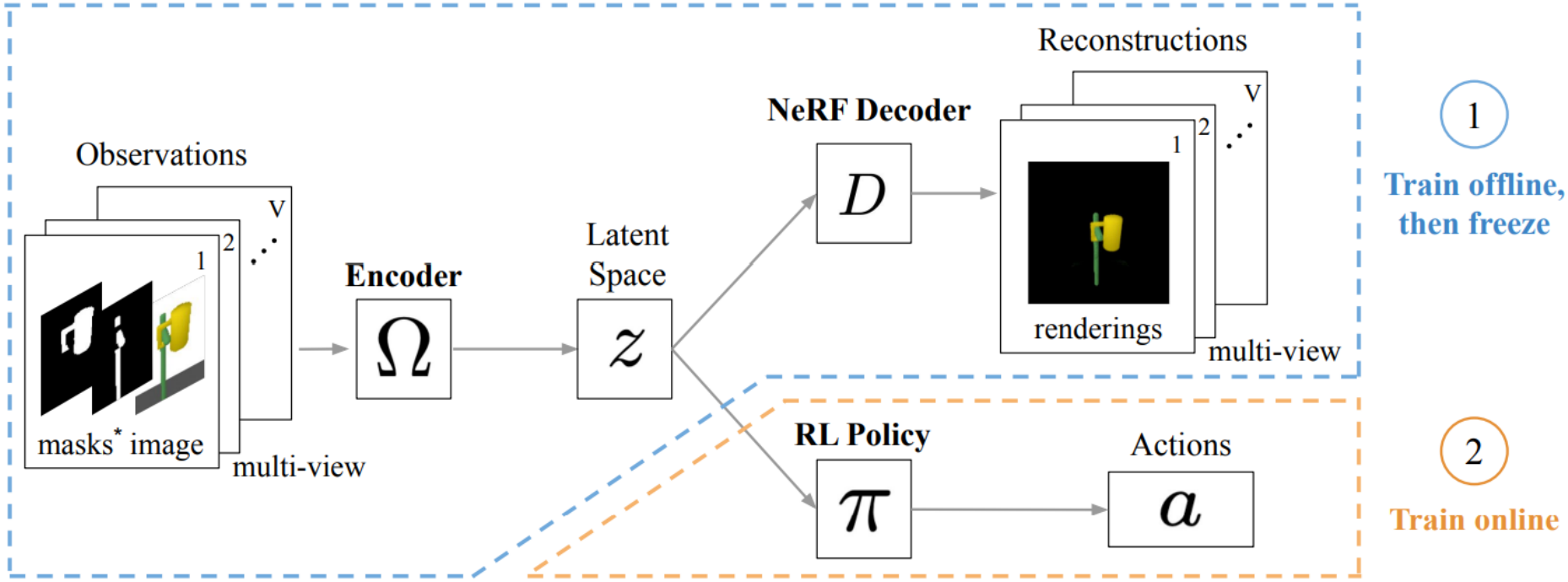
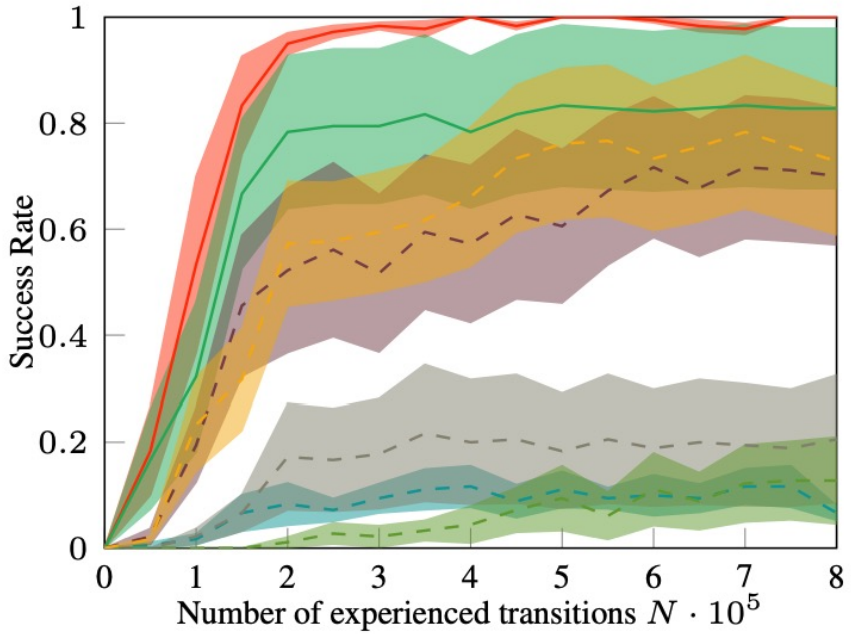


Figure 1: State representation learning for RL with NeRFs. First, the encoder and NeRF decoder are trained with supervision from a multi-view reconstruction loss on an offline dataset. Then, the encoder's weights are frozen, and the latent space is used as state input to train a policy with RL. *Masks of individual objects are only required for the compositional variant of our encoder.

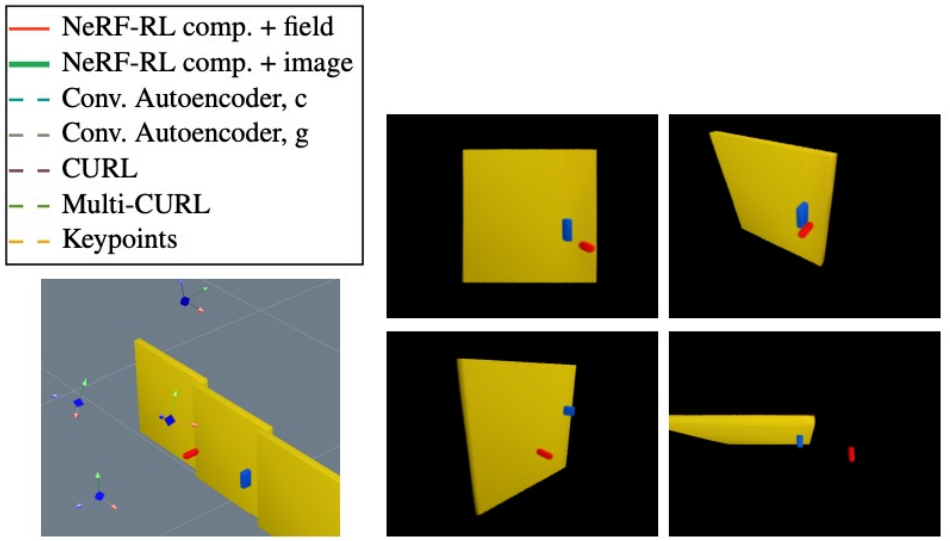
Figures are from Reinforcement Learning with Neural Radiance Fields

NeRF-RL

- Very efficient and effective!



(a) Learning curve.



(b) Left: Blue coordinate frames denote the four camera poses. Right: NeRF renderings.

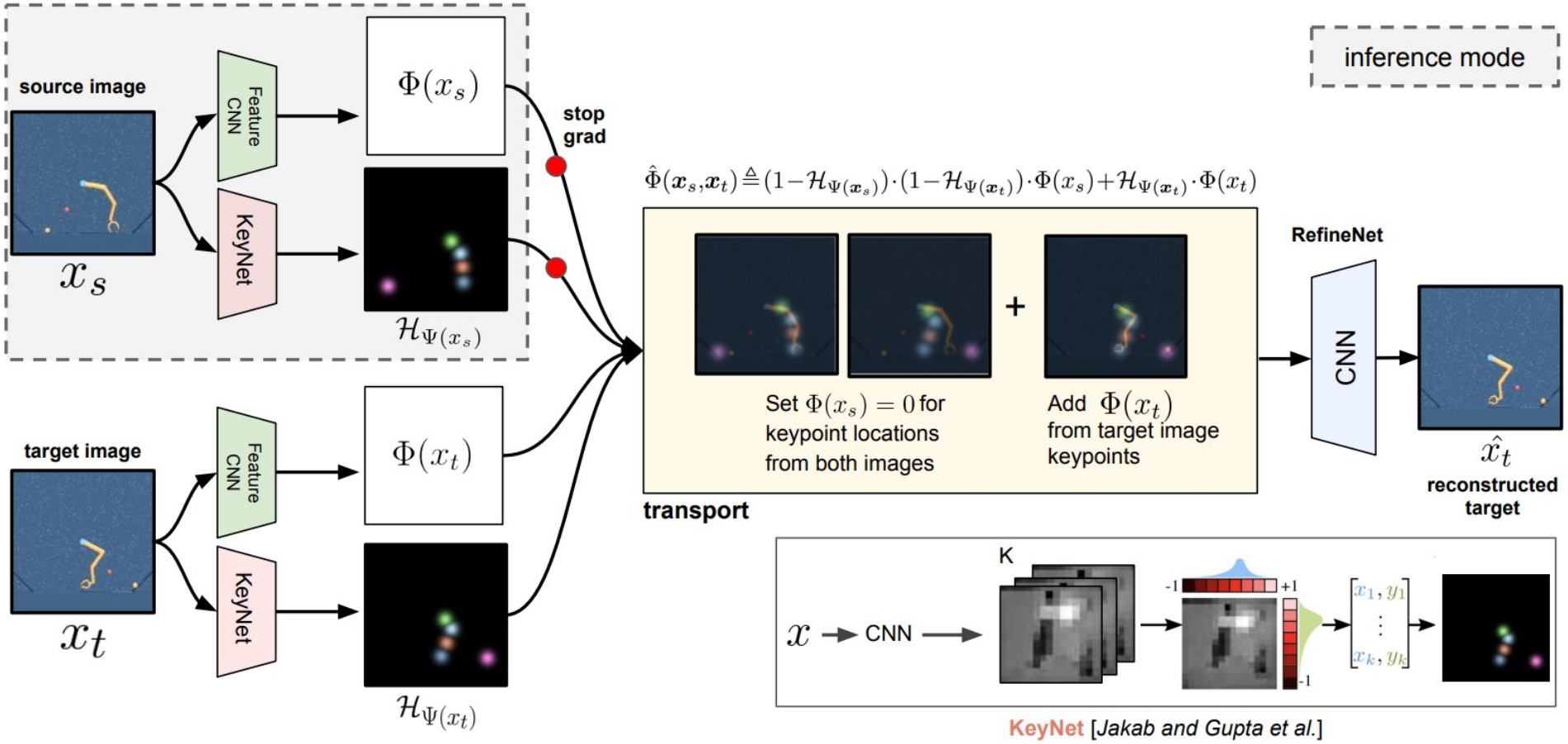
Figure 4: Door environment. (b) shows NeRF renderings for different scenes.

Figures are from Reinforcement Learning with Neural Radiance Fields

Pre-training for visual RL

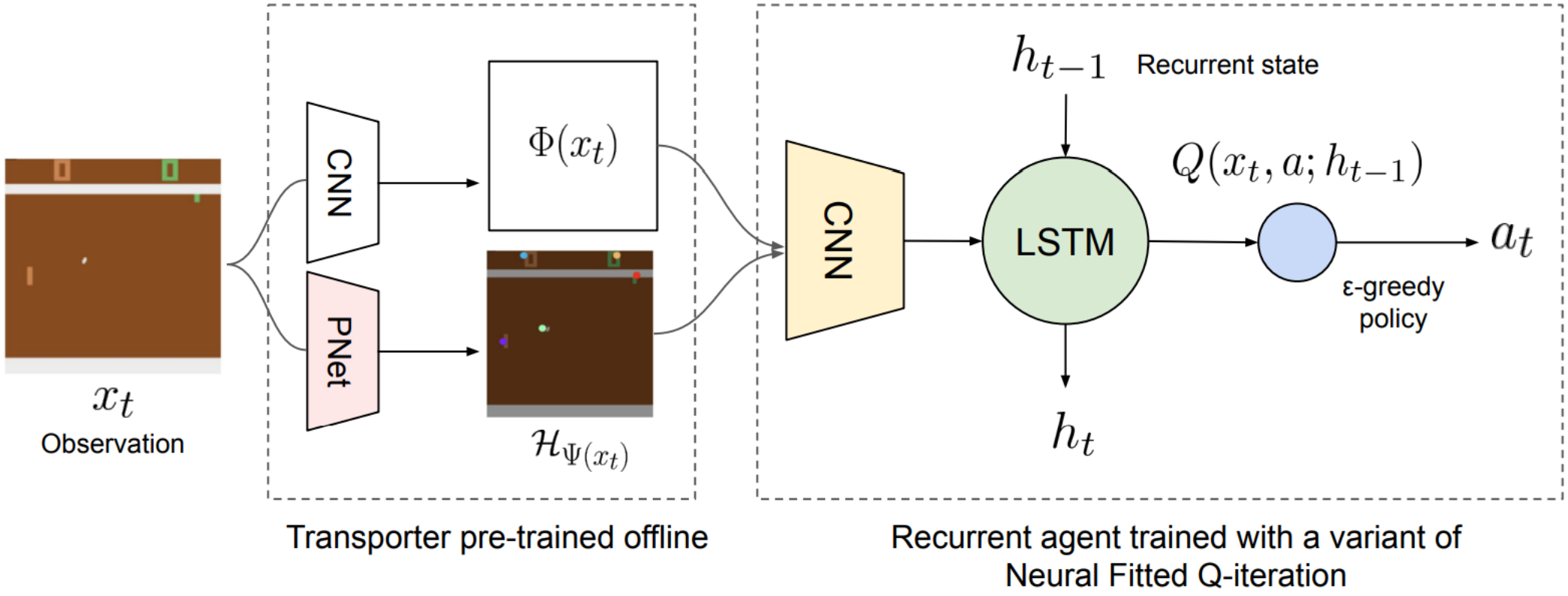
- Single-view pre-training
  - RRL
  - PVR
  - MVP
- Multi-view pre-training
  - NeRF-RL
- **Video pre-training**
  - Transporter
  - R3M
  - APV

Transporter



Figures are from Unsupervised learning of object key- points for perception and control

Transporter



Figures are from Unsupervised learning of object key- points for perception and control

R3M

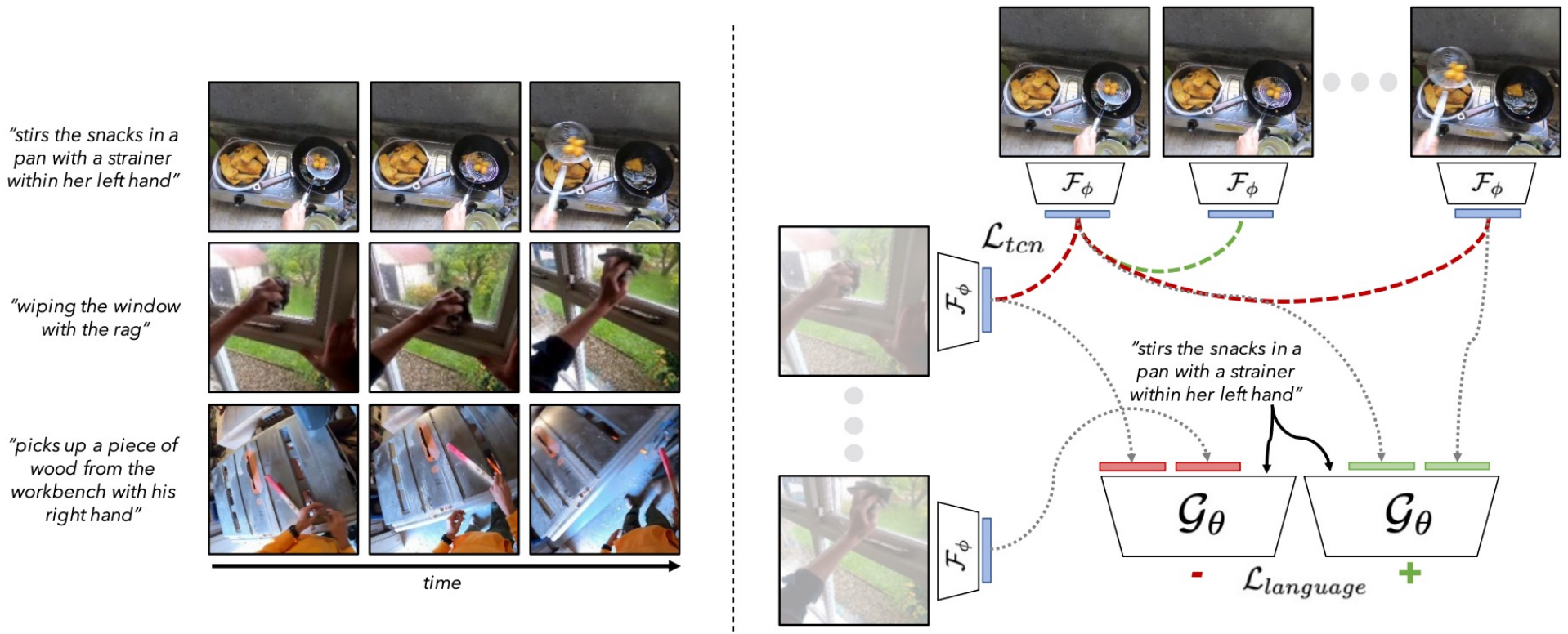


Figure 2: **Ego4D [16] Video and Language (left)**. Sample frames and associated language from Grauman et al. [16] used for training R3M. **R3M Training (right)**. We train R3M with time contrastive learning, encouraging states closer in time to be closer in embedding space and video-language alignment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

Figures are from R3M: A Universal Visual Representation for Robot Manipulation

R3M

- R3M is better than models from other pre-training techniques.

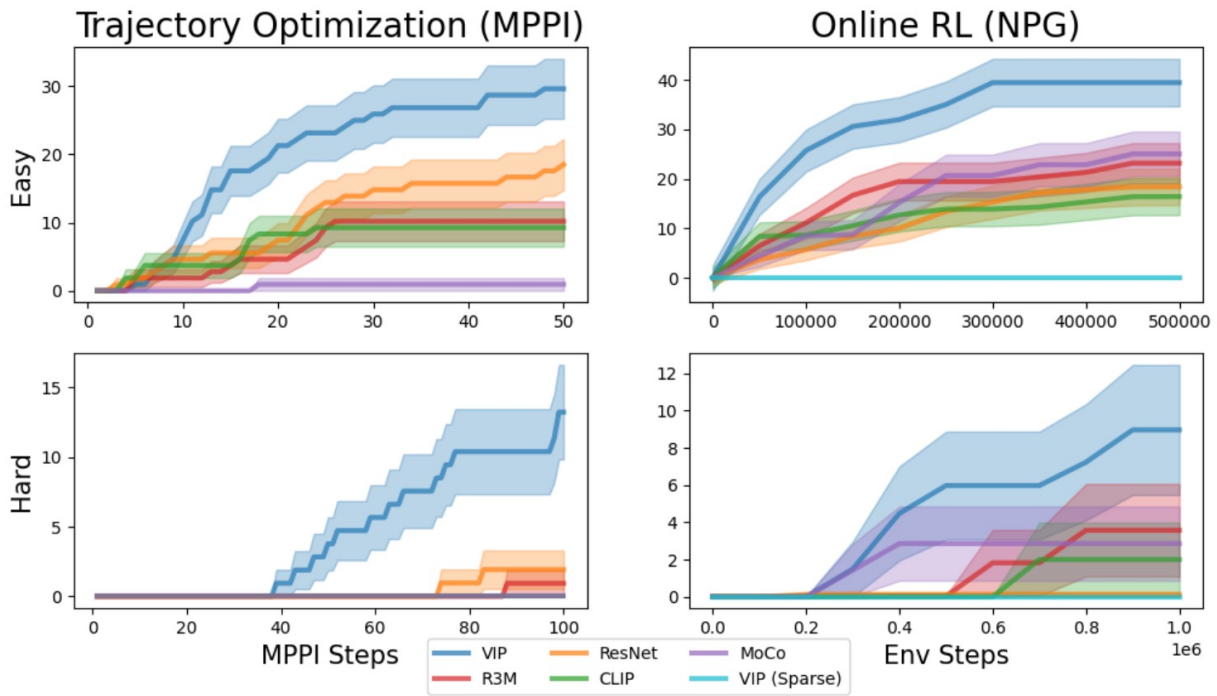
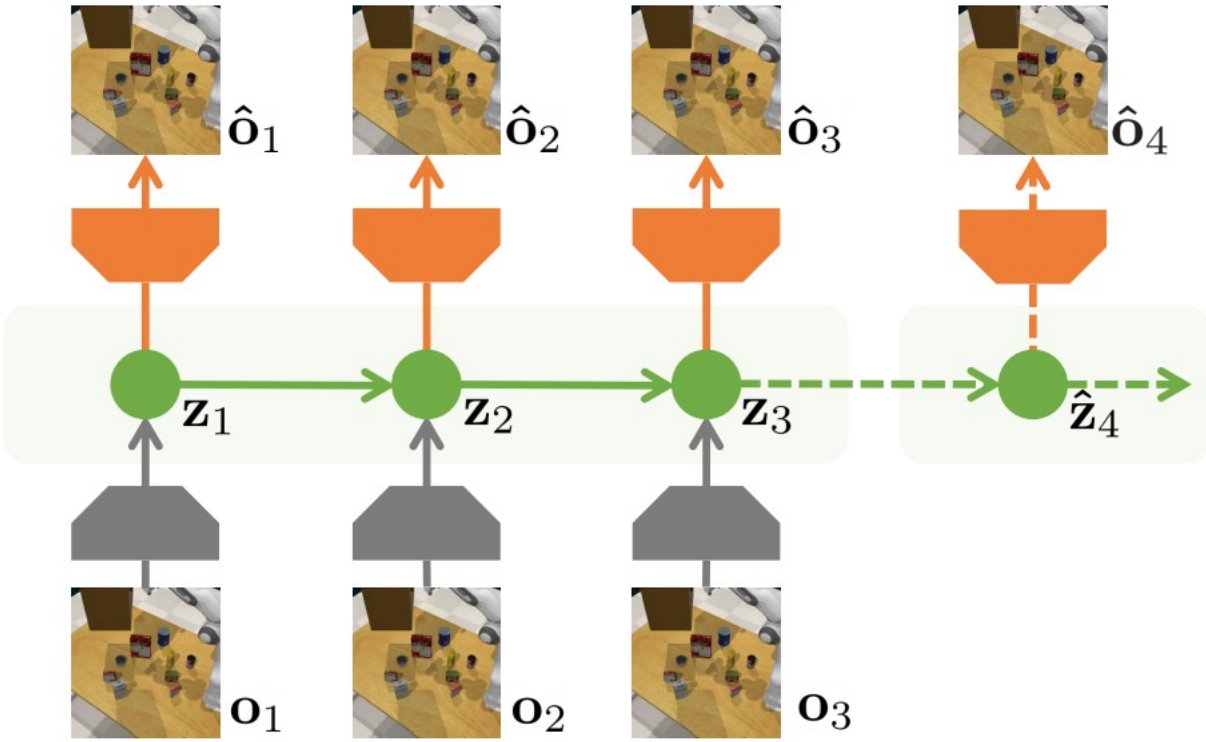


Figure 4: Visual trajectory optimization and online RL aggregate results (cumulative success rate %).

Figures are from VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training

APV

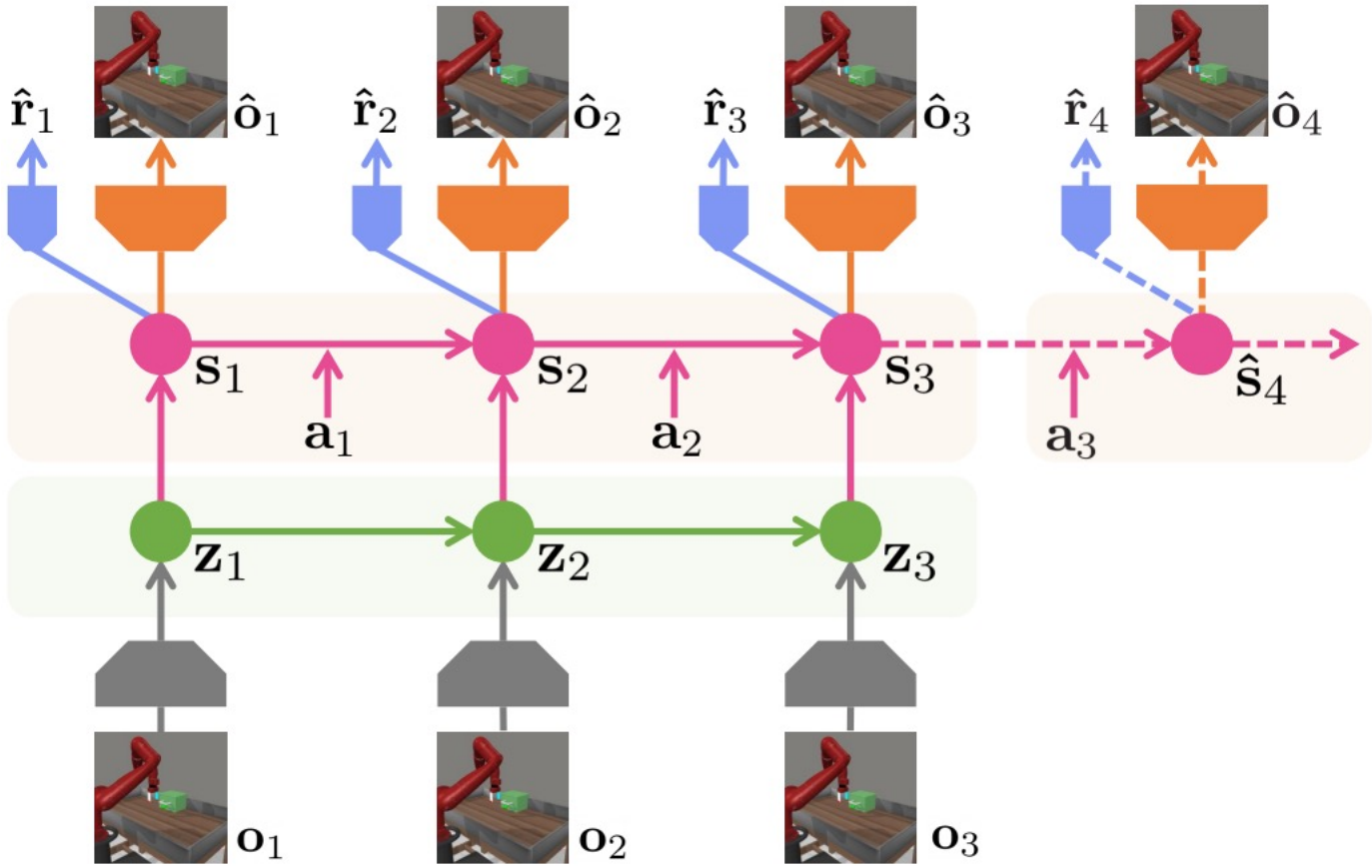
- Pre-train the image encoder over videos by reconstructing the observations without actions.
- Learn an image encoder that can extract features for dynamics learning.



Figures are from Reinforcement Learning with Action-Free Pre-Training from Videos

APV

- Learn a world-model over the pre-trained feature in the new tasks
- Optimize the agent with a model-based RL algorithm



Figures are from Reinforcement Learning with Action-Free Pre-Training from Videos

APV

- APV can be better than methods without pre-training.

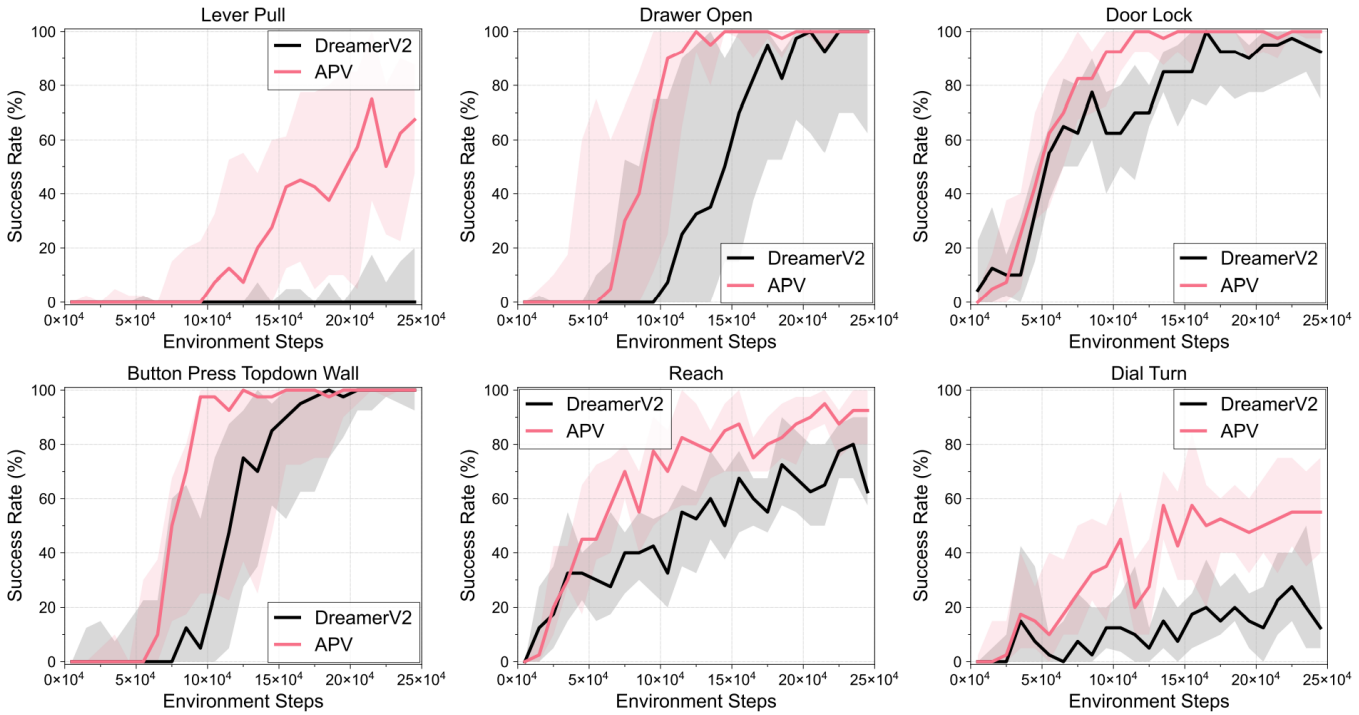


Figure 5. Learning curves on manipulation tasks from Meta-world as measured on the success rate. APV with generative pre-training on videos collected in manipulation tasks from RLBench consistently outperforms DreamerV2 in terms of sample-efficiency. The solid line and shaded regions represent the interquartile mean and bootstrap confidence intervals, respectively, across eight runs.

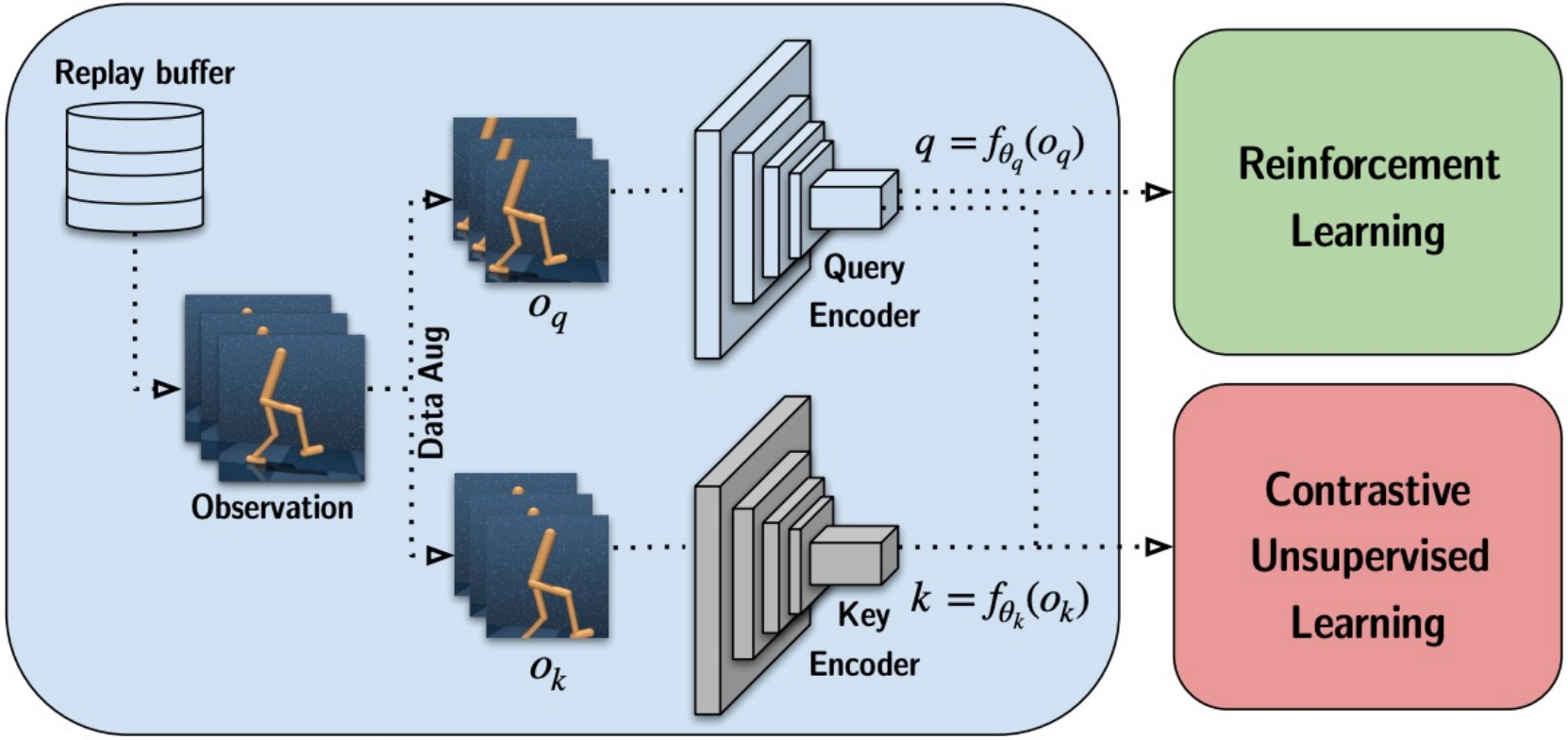
Figures are from Reinforcement Learning with Action-Free Pre-Training from Videos

Representation learning in visual RL

- **Self-supervised learning in visual RL**
  - CURL
  - MWM
- Data augmentation
  - DrQ
  - SVEA
  - DrQ-v2

CURL

- Add contrastive learning loss to learn better representation.



Figures are from CURL: Contrastive Unsupervised Representations for Reinforcement Learning

Results of CURL

- CURL can even have better performance than model-based methods, e.g., PlaNet, Dreamer.

500K step scores	CURL	PlaNet	Dreamer	SAC+AE	SLACv1	Pixel SAC	State SAC
Finger, spin	**926 ± 45**	561 ± 284	796 ± 183	884 ± 128	673 ± 92	179 ± 166	923 ± 21
Cartpole, swingup	**841 ± 45**	475 ± 71	762 ± 27	735 ± 63	-	419 ± 40	848 ± 15
Reacher, easy	**929 ± 44**	210 ± 390	793 ± 164	627 ± 58	-	145 ± 30	923 ± 24
Cheetah, run	518 ± 28	305 ± 131	570 ± 253	550 ± 34	**640 ± 19**	197 ± 15	795 ± 30
Walker, walk	**902 ± 43**	351 ± 58	897 ± 49	847 ± 48	842 ± 51	42 ± 12	948 ± 54
Ball in cup, catch	**959 ± 27**	460 ± 380	879 ± 87	794± 58	852 ± 71	312± 63	974 ± 33
100K step scores							
Finger, spin	**767 ± 56**	136 ± 216	341 ± 70	740 ± 64	693 ± 141	179 ± 66	811±46
Cartpole, swingup	**582±146**	297±39	326±27	311±11	-	419±40	835±22
Reacher, easy	**538±233**	20±50	314±155	274±14	-	145±30	746±25
Cheetah, run	299 ±48	138±88	235± 137	267±24	**319±56**	197±15	616±18
Walker, walk	**403±24**	224±48	277±12	394±22	361±73	42±12	891±82
Ball in cup, catch	**769 ± 43**	0 ± 0	246 ± 174	391± 82	512 ± 110	312± 63	746±91

Figures are from CURL: Contrastive Unsupervised Representations for Reinforcement Learning

MWM

- Self-supervised learning can improve model-based methods.
- Combine MAE with Dreamer-v2

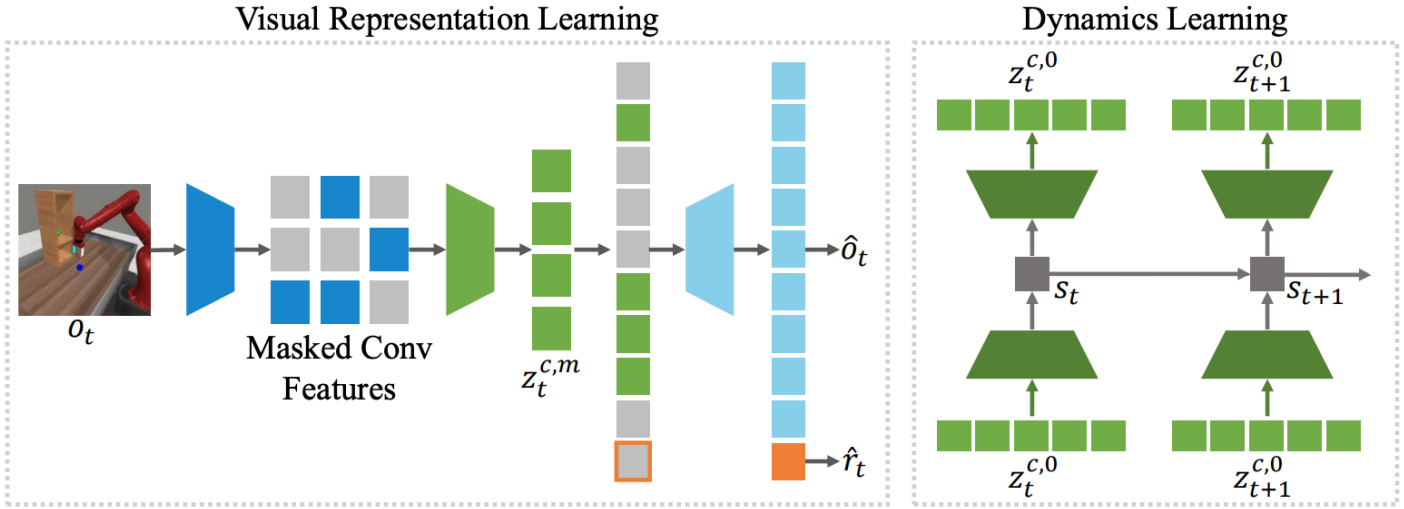


Figure 1: Illustration of our approach. We continually update visual representations and dynamics using online samples collected from environment interaction, by repeating iterative processes of training (Left) an autoencoder with convolutional feature masking and reward prediction and (Right) a latent dynamics model in the latent space of the autoencoder. We note that autoencoder parameters are not updated during dynamics learning.

Figures are from Masked World Models for Visual Control

Representaition learning in visual RL

- Self-supervised learning in visual RL
  - CURL
  - MWM
- **Data augmentation**
  - DrQ
  - SVEA
  - DrQ-v2

DrQ

- Train critic with augmented observations.
- Target value is the average value computed with K different augmentations.
- The critic loss will encourage the value computed with augmented observations to be close the target value.

Algorithm 1 DrQ: **D**ata-**r**egularized **Q** applied to a generic off-policy actor critic algorithm.
Black: unmodified off-policy actor-critic.
Orange: image transformation.
Green: target Q augmentation.
Blue: Q augmentation.

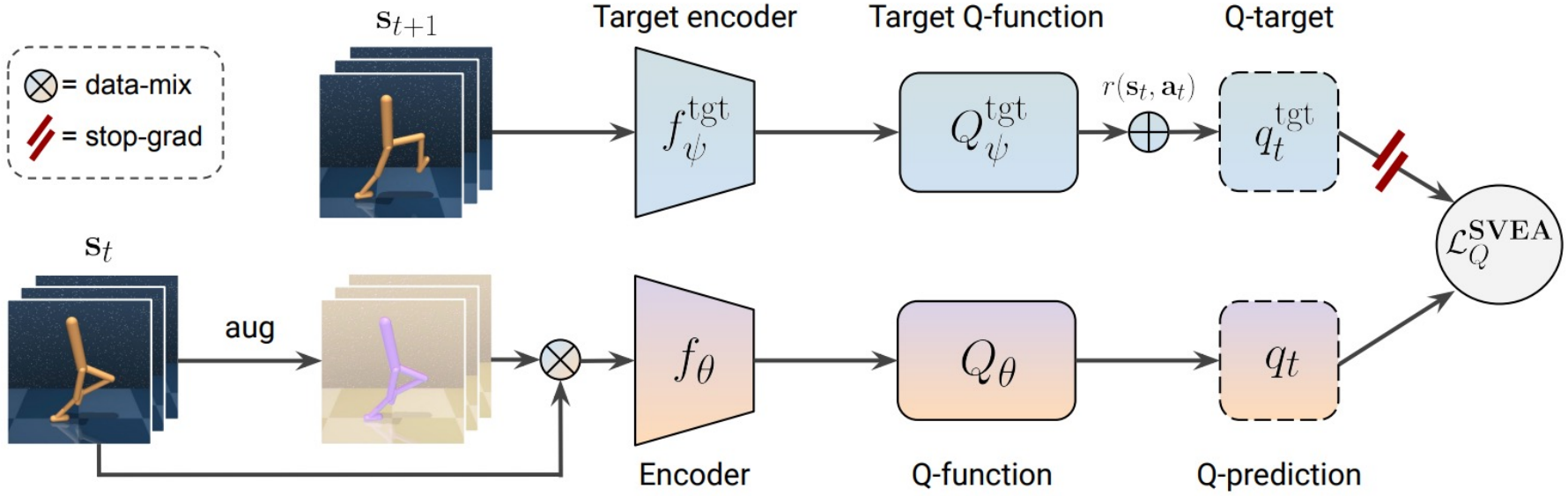
Hyperparameters: Total number of environment steps T, mini-batch size N, learning rate λ_θ, target network update rate τ, image transformation f, number of target Q augmentations K, number of Q augmentations M.

for each timestep $t = 1..T$ **do**
  $a_t \sim \pi(\cdot|s_t)$
  $s'_t \sim p(\cdot|s_t, a_t)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r(s_t, a_t), s'_t)$
  UPDATECRITIC($\mathcal{D}$)
  UPDATEACTOR($\mathcal{D}$) ▷ Data augmentation is applied to the samples for actor training as well.
end for
procedure UPDATECRITIC($\mathcal{D}$)
  $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^N \sim \mathcal{D}$ ▷ Sample a mini batch
  $\left\{\nu'_{i,k} \middle| \nu'_{i,k} \sim \mathcal{U}(\mathcal{T}), i = 1..N, k = 1..K\right\}$ ▷ Sample parameters of target augmentations
  for each $i = 1..N$ **do**
    $a'_i \sim \pi(\cdot|s'_i)$ or $a'_{i,k} \sim \pi(\cdot|\, f(s'_i, \nu'_{i,k})), k = 1..K$
    $\hat{Q}_i = Q_{\theta'}(s'_i, a'_i)$ or $\hat{Q}_i = \frac{1}{K}\sum_{k=1}^K Q_{\theta'}(f(s'_i, \nu'_{i,k}), a'_{i,k})$
    $y_i \leftarrow r(s_i, a_i) + \gamma \hat{Q}_i$
  end for
  $\{\nu_{i,m}|\nu_{i,m} \sim \mathcal{U}(\mathcal{T}), i = 1..N, m = 1..M\}$ ▷ Sample parameters of Q augmentations
  $J_Q(\theta) = \frac{1}{N}\sum_{i=1}^N (Q_\theta(s_i, a_i) - y_i)^2$ or $J_Q(\theta) = \frac{1}{NM}\sum_{i,m=1}^{N,M}(Q_\theta(f(s_i, \nu_{i,m}), a_i) - y_i)^2$
  $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta J_Q(\theta)$ ▷ Update the critic
  $\theta' \leftarrow (1-\tau)\theta' + \tau\theta$ ▷ Update the critic target
end procedure

Figures are from Regularizing Deep Reinforcement Learning from Pixels

SVEA

- Adding augmentation when computing the target value will
  - introduce noise
  - make the training not stable.
- Solution: do not augment the observations for target value!



Figures are from Stabilizing Deep Q-Learning with ConvNets and Vision Transformers under Data Augmentation

DrQ-v2

- Improvement of DrQ-v2 from DrQ:
  - Use DDPG as the base algorithm instead of SAC
  - Use multi-step return.
  - Use a larger capacity of the replay buffer.
  - Add bilinear interpolation to the random shift image augmentation.
  - Introduce an exploration schedule.

DrQ-v2 can solve more challenging tasks

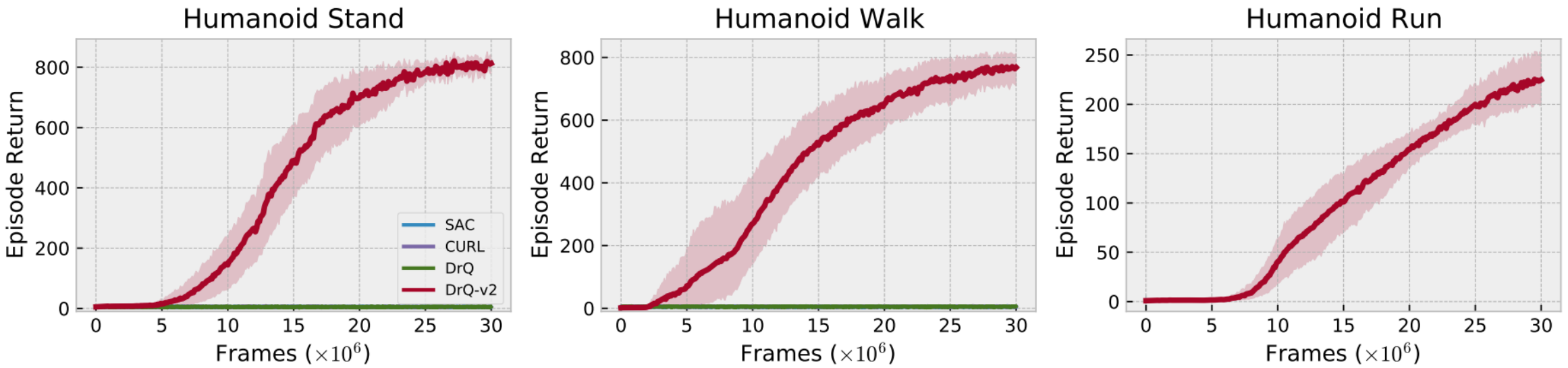


Figure 3: The *hard* benchmark consists of three humanoid locomotion tasks: *stand*, *walk*, and *run*. These three represent particularly hard exploration challenges, being previously unsolvable by model-free methods. The training speed of DrQ-v2 was key to solving the task, since it allowed for extensive investigation of different variations, resulting in the discovery of effective strategies.

Figures are from Improved Data-Augmented Reinforcement Learning

Visual RL in the Real World

Deploy visual RL agents in real-world applications

Two strategies of training visual RL in the real world

- **Train visual RL directly in the real world**
  - QT-Opt
  - AW-Opt
  - MT-Opt
- Sim2Real
  - RL-CycleGAN
  - RetinaGAN

QT-Opt

- Initialize the replay buffer with 580K off-policy samples collected with 7 robots in several weeks.
- The action is selected by using CEM on the learn Q function.
- The agent is further finetuned online with a sparse reward.

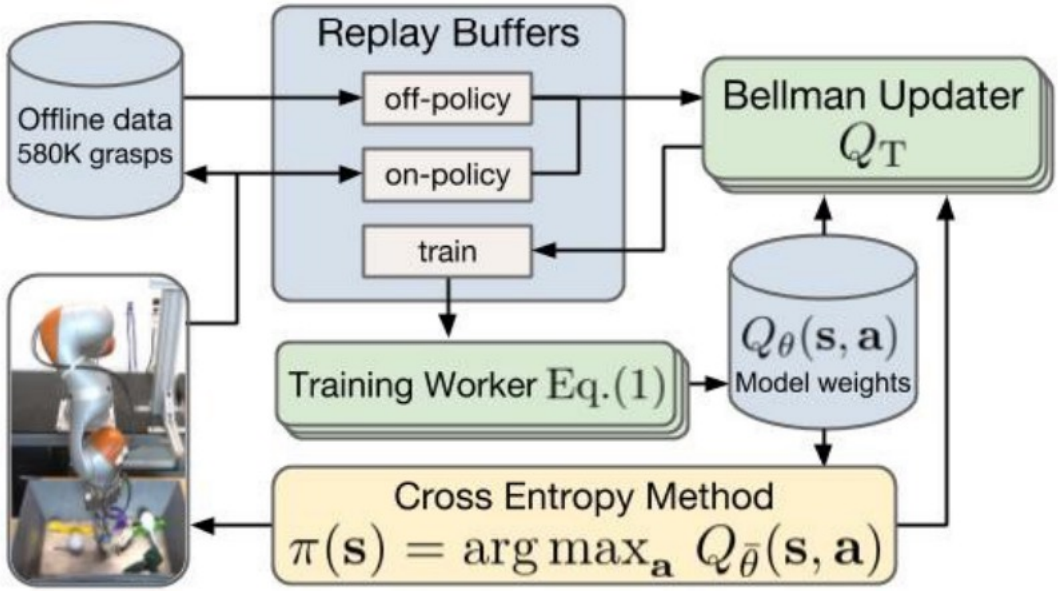


Figure 3: Our distributed RL infrastructure for QT-Opt (see Sec. 4.2). State-action-reward tuples are loaded from an offline data stored and pushed from online real robot collection (see Sec. 5). Bellman update jobs sample transitions and generate training examples, while training workers update the Q-function parameters.

Figures are from QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation

QT-Opt

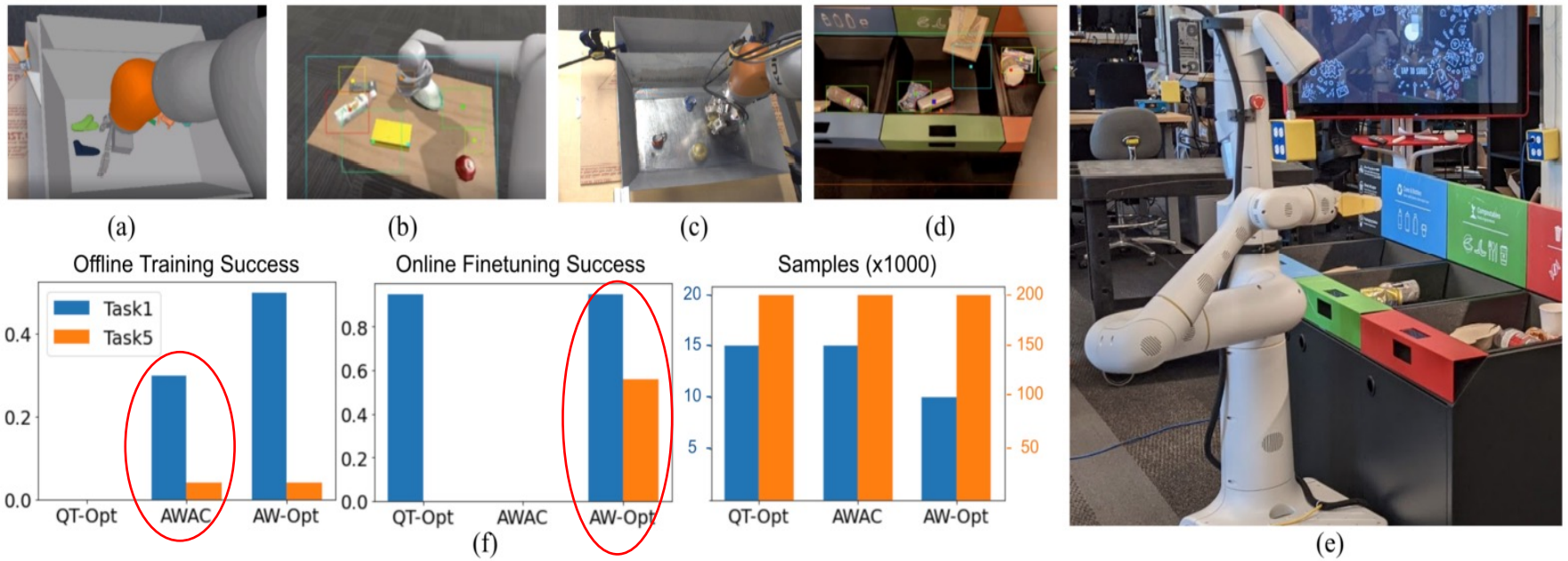
- Use image subtraction to know if the object is moved to the box.
- Reward design in real world visual RL is very tricky!
  - How to define the reward in the pixel space?
- Full training time is about **four months**!

Figure 12: Grasp success is determined by subtracting images before an object is dropped into the bin (left) and after it was dropped (right).

Figures are from QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation

AW-Opt

- Pre-training with offline RL, e.g., AWAC, and offline data.
- Online finetune with visual RL



Figures are from AW-Opt: Learning Robotic Skills with Imitation and Reinforcement at Scale

MT-Opt

- Scale up QT-Opt to multiple tasks with multi-task data sharing.

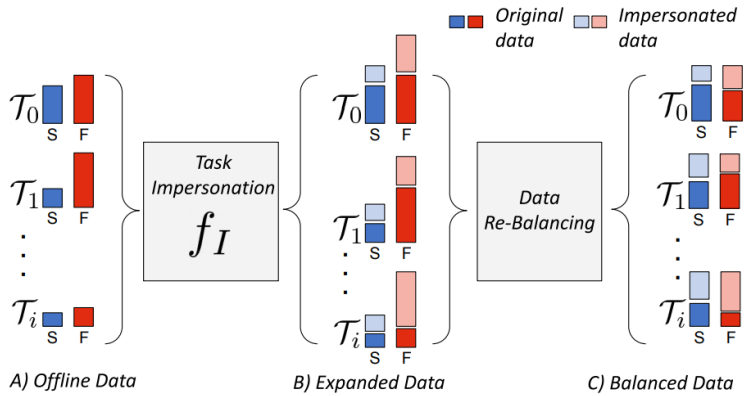
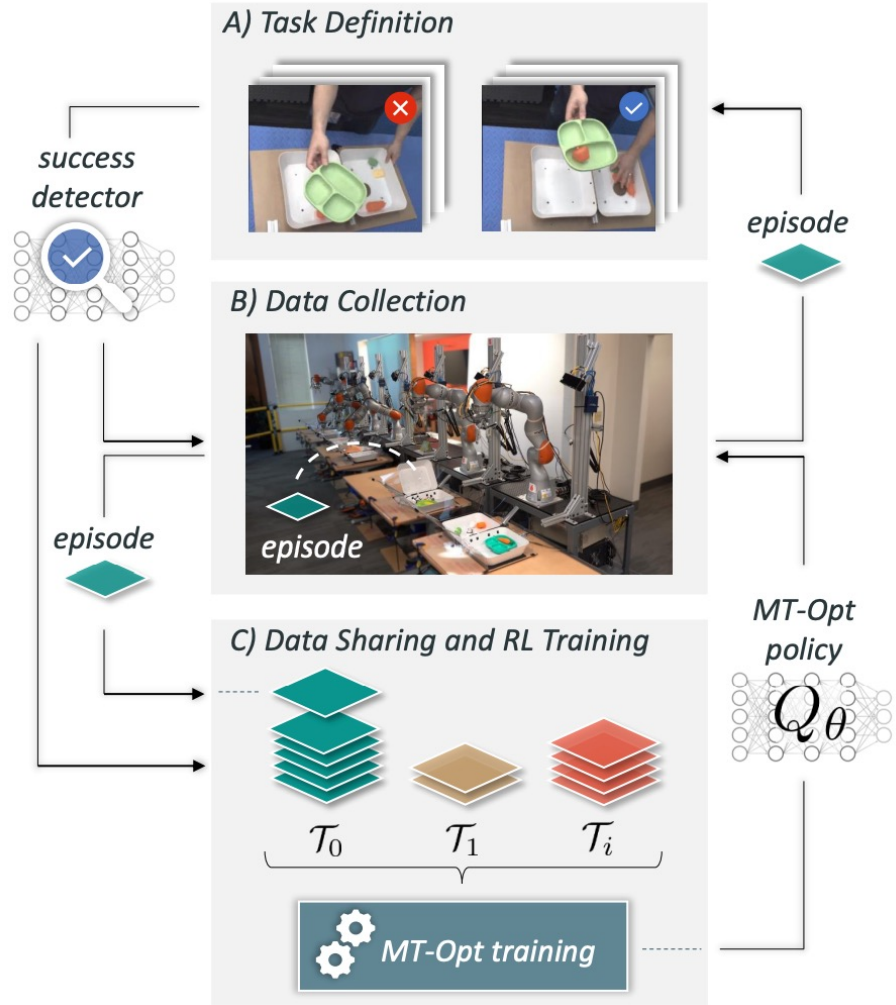


Fig. 3: Path of episodes through task impersonation, where episodes are routed to train relevant tasks, and data re-balancing where the ratio of success (S) and failure (F) episodes and proportion of data per task is controlled. Pale blue and pale red indicates additional task training data coming from other tasks. The height of a bar indicates very different amount of data across tasks and across successful outcomes.



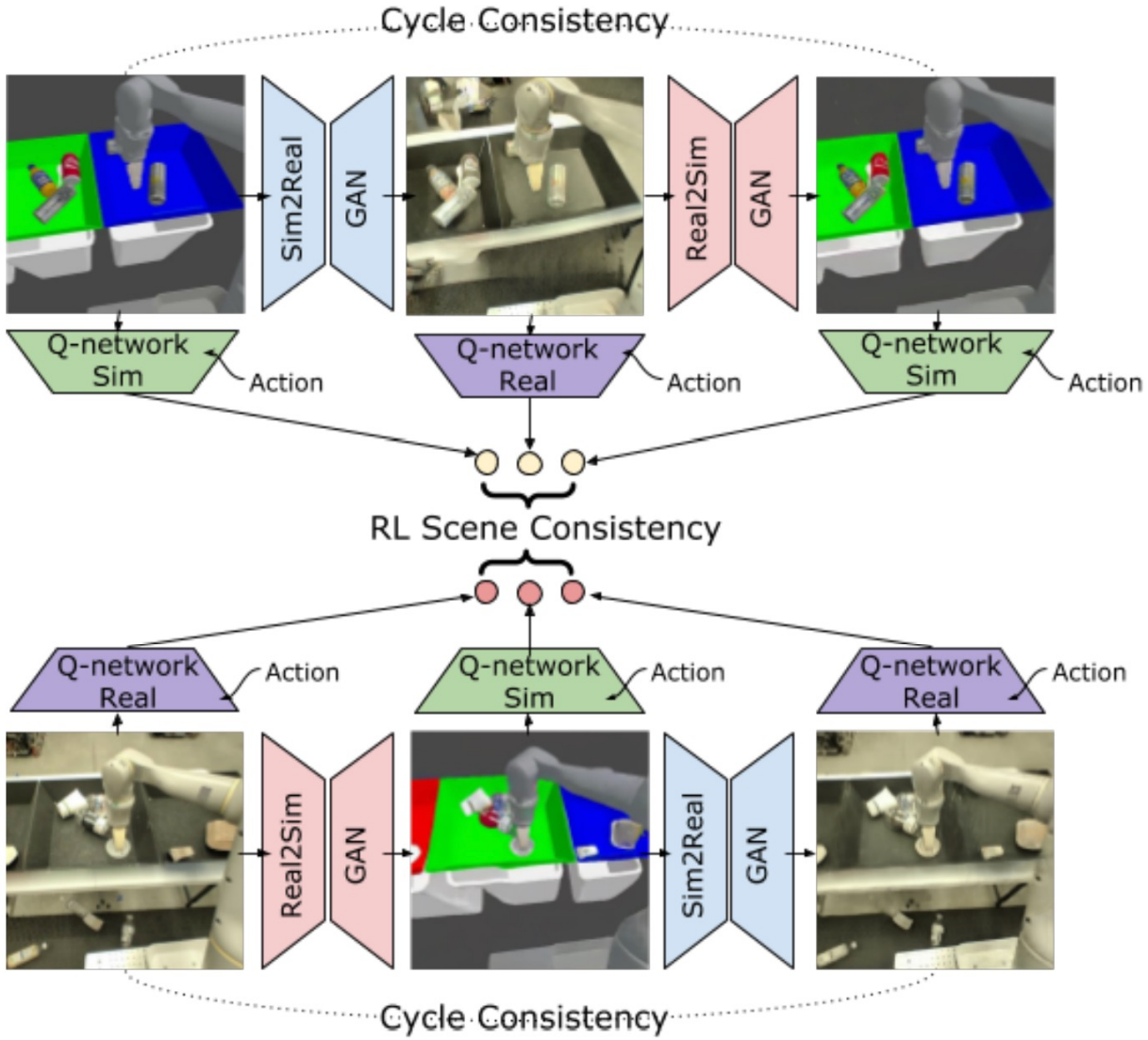
Figures are from MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale

Two strategies of training visual RL in the real world

- Train visual RL directly in the real world
  - QT-Opt
  - AW-Opt
  - MT-Opt
- **Sim2Real**
  - RL-CycleGAN
  - RetinaGAN

RL-CycleGAN

- Two image GANs Sim2Real and Real2Sim are trained with cycle consistency loss.
- RL scene consistency loss: encouraging cycle consistency over Q networks.

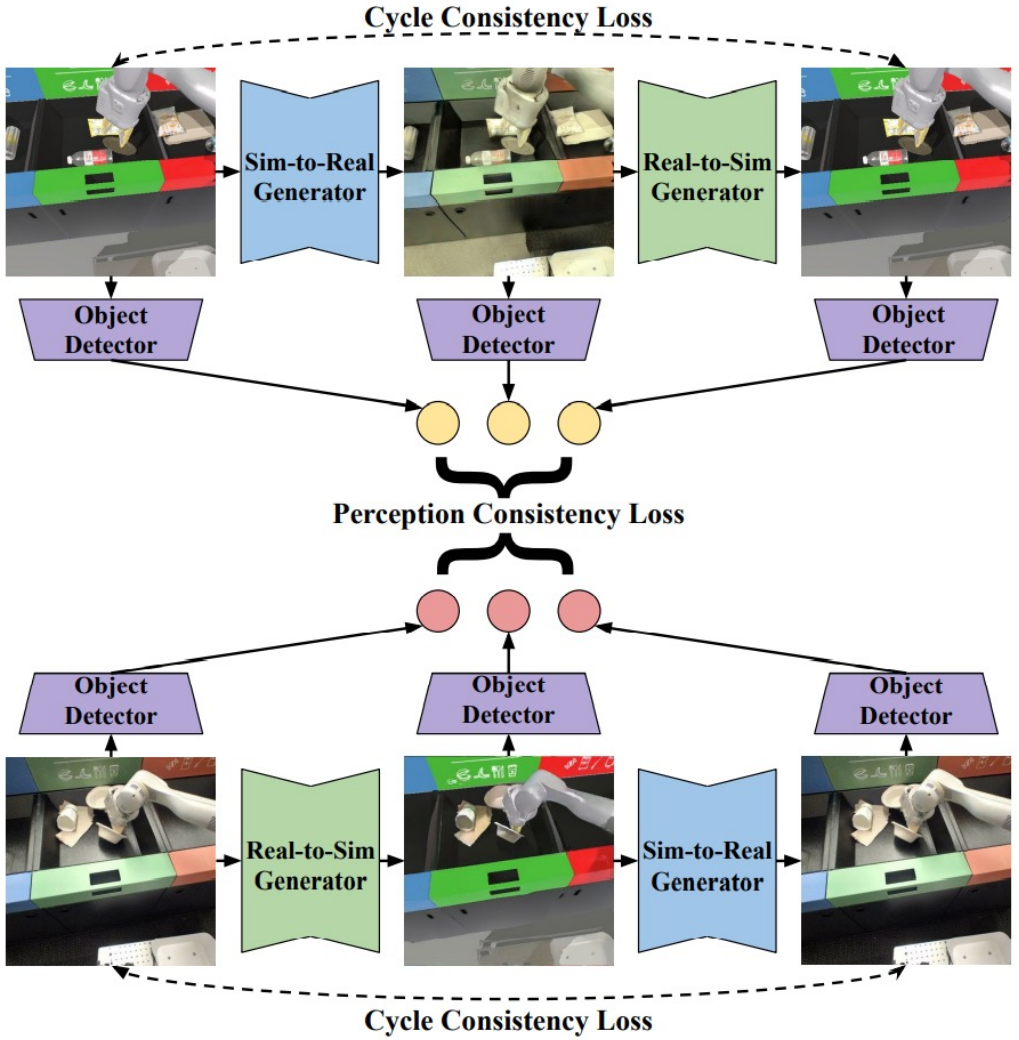


Figures are from RL-CycleGAN: Reinforcement Learning Aware Simulation-To-Real

RetinaGAN

- Add cycle consistency loss over perception tasks, e.g., detection.

Model	Grasp Success	Est. Std.
Sim-Only	18.9%	4.1%
Randomized Sim	41.1%	5.2%
GAN: 10K Real, Q2-Opt: 10K Real		
Real	22.2%	4.4%
RetinaGAN	47.4%	5.3%
RetinaGAN+Real	**65.6%**	5.0%
GAN: 135K Real, Q2-Opt: 211K Real		
Real	30.0%	4.9%
Sim+Real	54.4%	5.3%
RetinaGAN+Real	**80.0%**	4.2%
GAN: 135K Real, Q2-Opt: 0 Real		
CycleGAN [5]	67.8%	5.0%
RL-CycleGAN [24]	68.9%	4.9%
RetinaGAN	**80.0%**	4.2%



Figures are from RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer

Future directions

Direction1-3D vision in visual RL

- 3D representations, e.g., meshes, point clouds, voxels, are widely used in computer vision applications, e.g., object detection in autonomous driving.
- Can 3D representations improve the sample efficiency and performance of visual RL?
- Can visual RL agents learned on 3D inputs generalize better?
- Are representation learning techniques still helpful for visual RL with 3D observations?
- Can the agent learn world-models more easily over 3D observations?

Direction2-Visual language RL

- Visual language models(VLMs) are **HOT** topics recently.
- Some recent works begin to explore this direction!
  - Use VLM models as pre-trained feature extractors in visual RL.
- How to use VLMs to help training visual RL agent?
  - Exploration? Generalization?
- How to construct a proper benchmark for evaluating visual-language RL?

Research Progress

- Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations (NeurIPS 2021 Dataset) Co-first author
- Frame Mining: a Free Lunch for Learning Robotic Manipulation from 3D Point Clouds (CoRL 2022) Co-first author
- State alignment-based imitation learning (ICLR 2020)
- Improving policy optimization with generalist-specialist learning (ICML 2022)
- Approximate Convex Decomposition for 3D Meshes with Collision-Aware Concavity and Tree Search(SIGGRAPH 2022)
- In submission:
- Close the Visual Domain Gap by Physics-Grounded Active Stereovision Depth Sensor Simulation (to T-RO)
- ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills (to ICLR 2023)
- Variational Reparametrized Policy Learning with Differentiable Physics (to ICLR 2023)

Thanks!