CLVR Implementation Project:

Reward-Induced Representation Learning [1]

Yibin Xiong

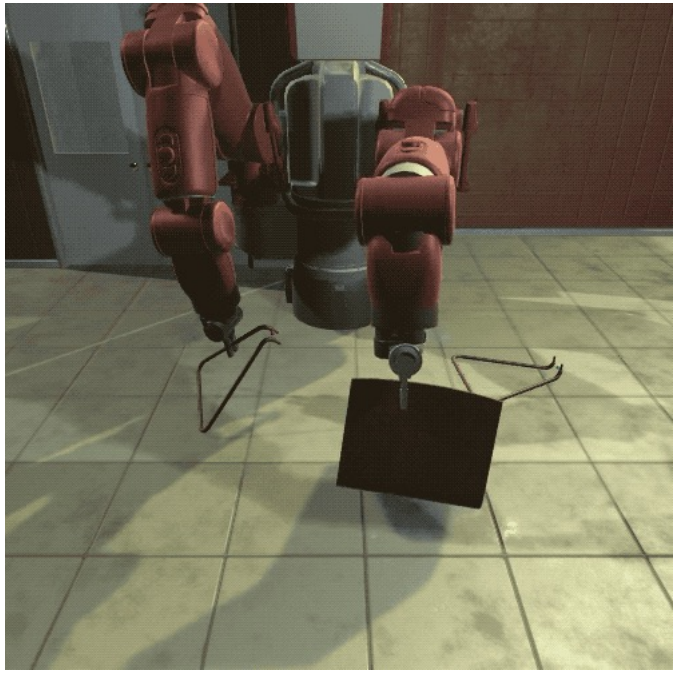
October 14, 2021

Outline

- Overview
- Experiment Details
- Results & Analysis
- Reflections & Implications

Motivation

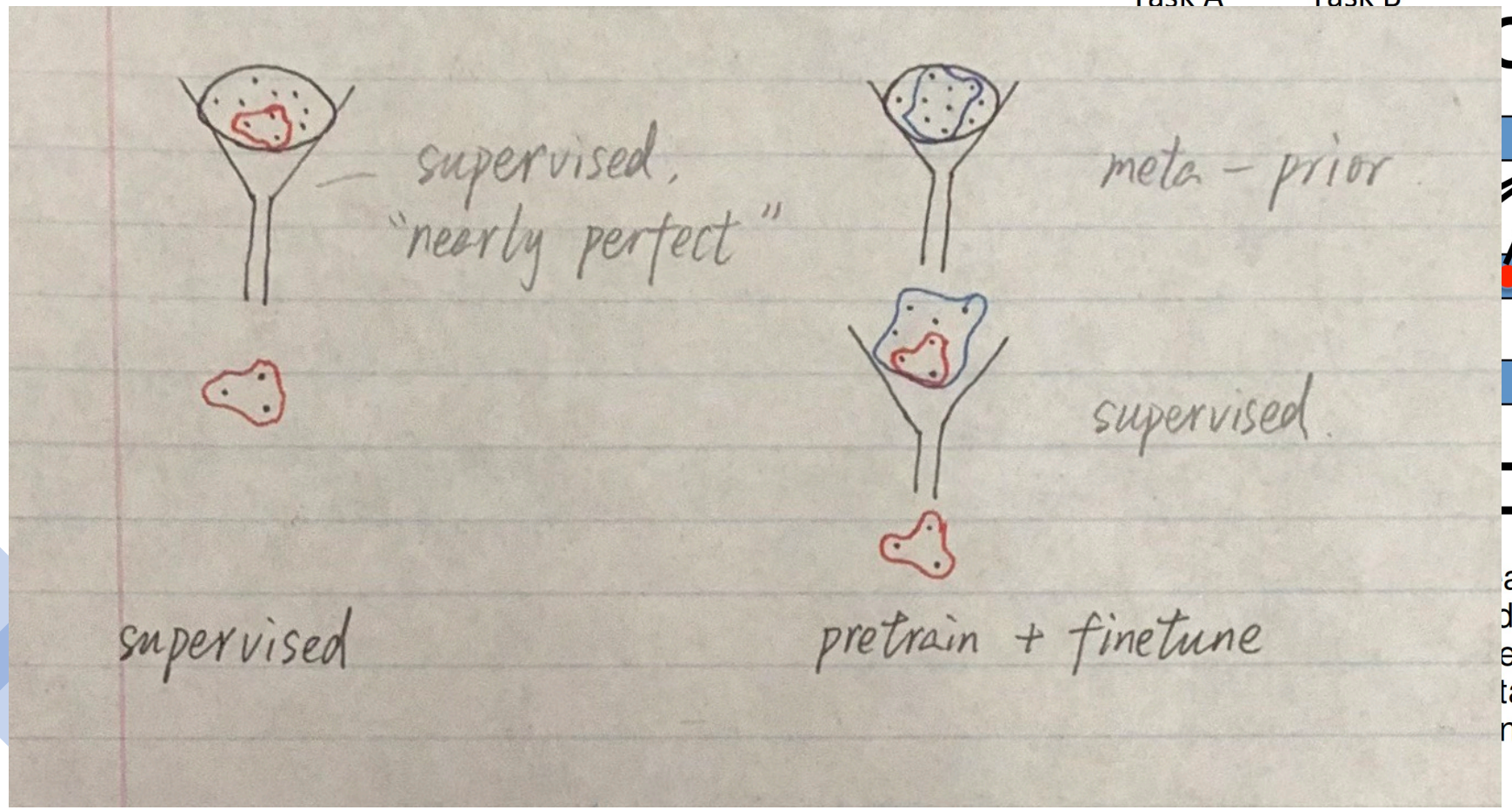
- High dimensional input
- Sparse reward (most actions result in 0 reward)

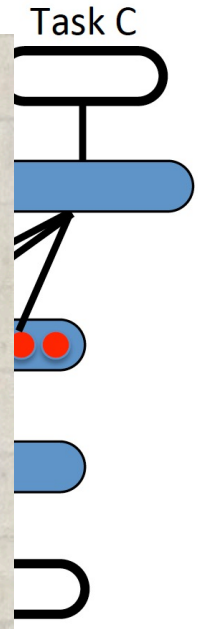


- Extract *useful* information, ignore *useless* information

Representation Learning

- Idea: find a lower-dimensional representation (encoding/feature vector) of the input that retains important information relevant to our tasks
- Think about filters!





arning model which
dden layer, in red),
ervised setting), and
task. Because these
ngth allows gains in

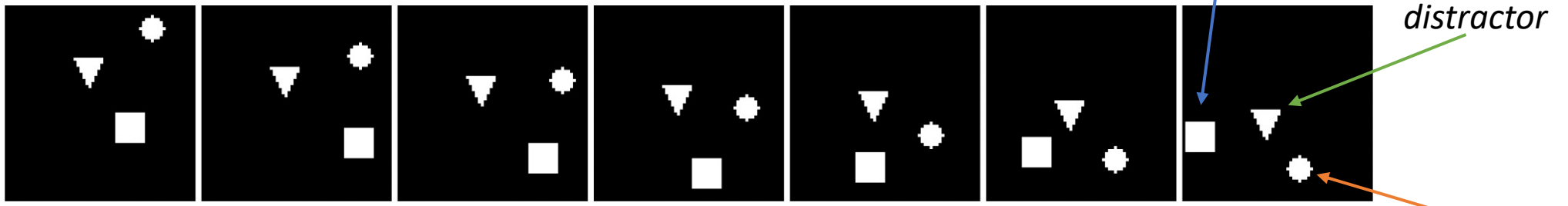
Our Scheme

Let $\mathcal{T}$ be the distribution of RL tasks of our interest and $\mathcal{D}$ be the pre-training dataset (i.e. a collection of reward-annotated trajectories, denoted as $\mathcal{D} = \{s^t, a^t, r^t_{1:K}, \dots\}$).

i) We sample some tasks $T_1, \dots, T_K$ from $\mathcal{T}$ to pre-train an encoder.
ii) Then we use the encoder to extract crucial information and conduct any task(s) in $\mathcal{T}$ based on it.

Our RL Environment: Sprites

- Sprites: a customized environment that generates sequences of multiple geometric objects in a 2D frame



- State: (x, y, v_x, v_y); Observation: 64*64 grayscale image tensors
- Pre-training tasks: reward is proportional to the (x,y) position of the *agent*. and target.
- Downstream task: *agent* is supposed to follow the *target*, and not misled by the *distractor*.

$$R = 1 - \frac{1}{\sqrt{2}} \| p_{target} - p_{agent} \|_2$$

Math Formulation

- We want to train a reward prediction model $p(r^t_{1:K}|x)$.
- Factoring out this probability, we get

$$p(r^t_{1:K}|x) = \prod_{k=1}^{K} \int p_{\eta_k}(r_k|z) \; p_\phi(z|x) dz$$

reward head encoder

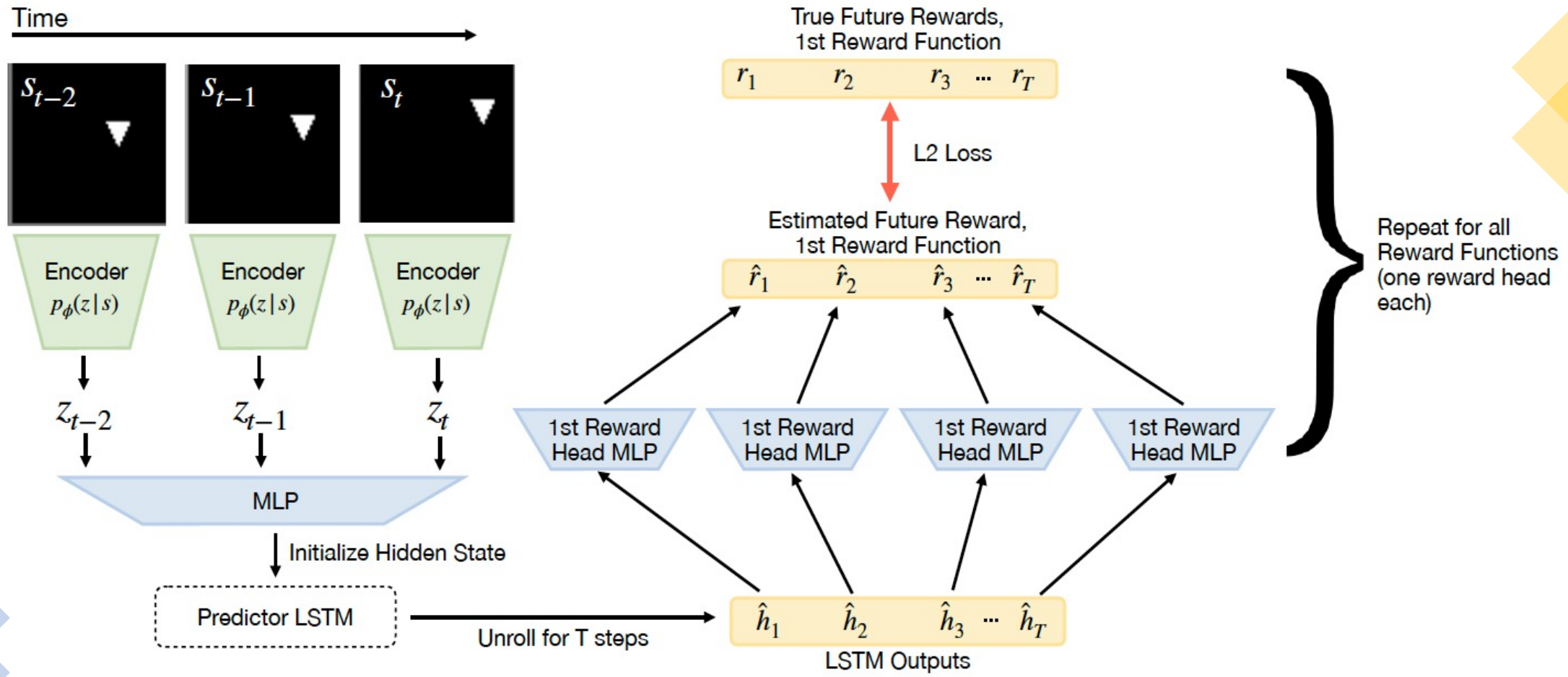
- Based on this probability distribution, we make a prediction $\hat{r}^t_{1:K}$ of the rewards.
- The parameters are $\{\phi, \eta_1 \dots \eta_K\}$. Every time we update the parameters to minimize the following loss function

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{k=1}^{K} \| r^t_k - \hat{r}^t_k \|^2$$

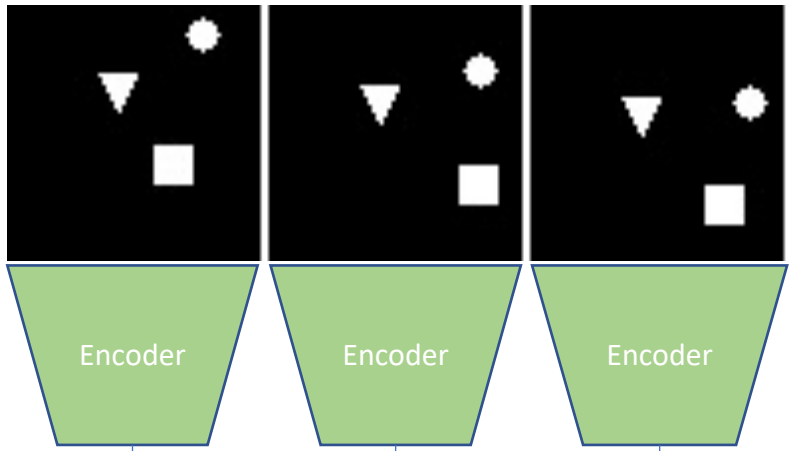
Outline

- Overview
- Experiment Details
- Results & Analysis
- Reflections & Implications

Reward-Prediction Model (Pre-training)

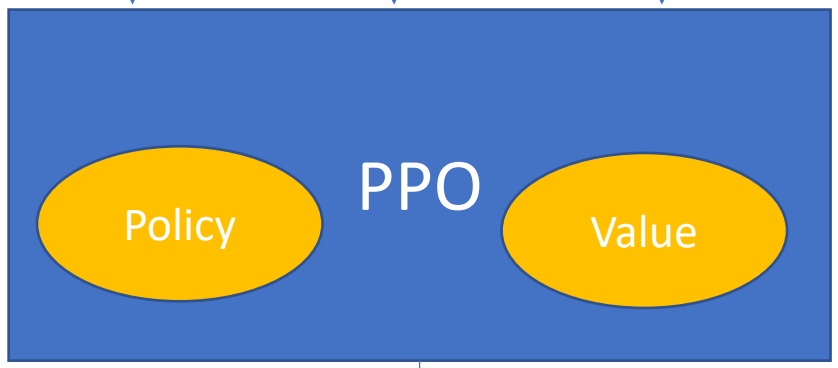


Downstream RL



trajectories in the *buffer*

$z_t \in R^{64}$
reward-induced representation



Good Policy $\pi(a|s)$

Baselines: Using an Encoder

- **Reward-prediction** is basically what we described. It uses reward-induced encoder

2 versions: If we update the parameters of the encoder in downstream training, it is called the *reward-prediction-finetune* baseline. If we freeze the encoder, it is the *reward-prediction* baseline.

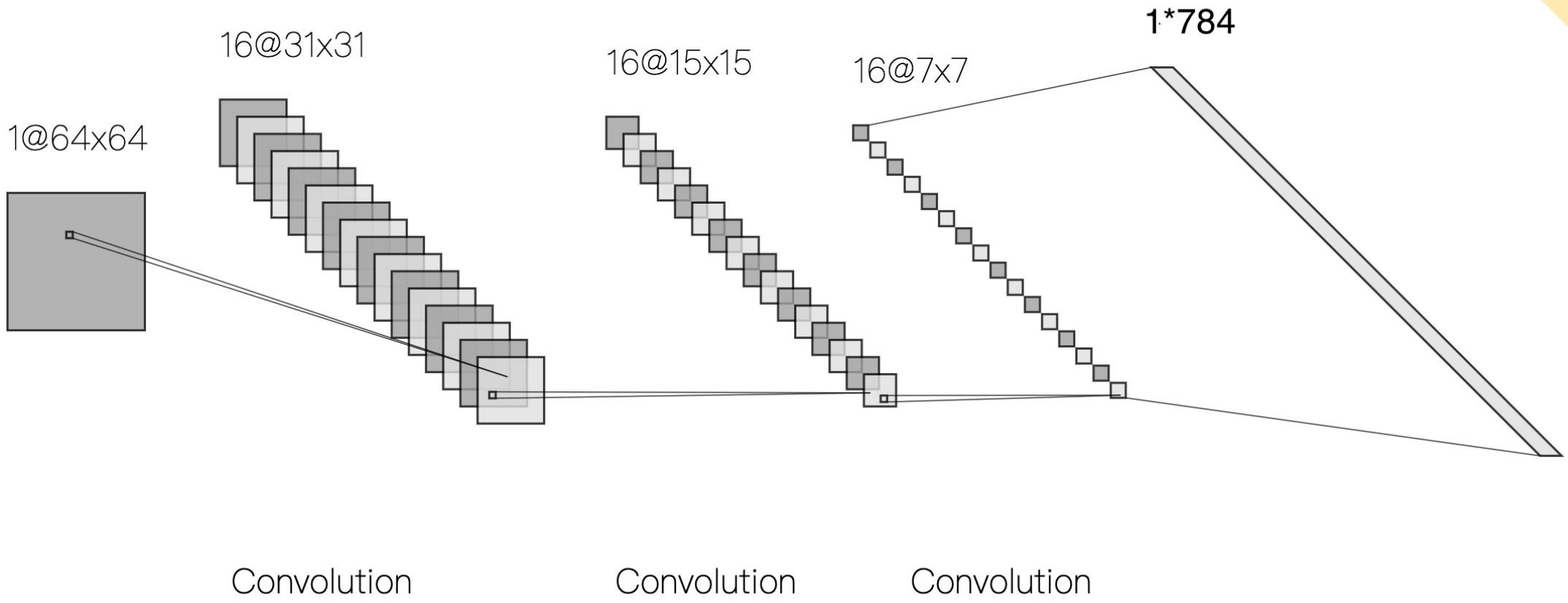
- **Image-reconstruction** uses the encoder trained with image reconstruction loss (pixel-wise L2 loss). Other parts are the same as reward-prediction.

Still, there are the finetuned and frozen versions.

- **Image-scratch** adopts the same architecture as *reward-prediction*, but reinitializes the parameters of the reward-induced encoder.

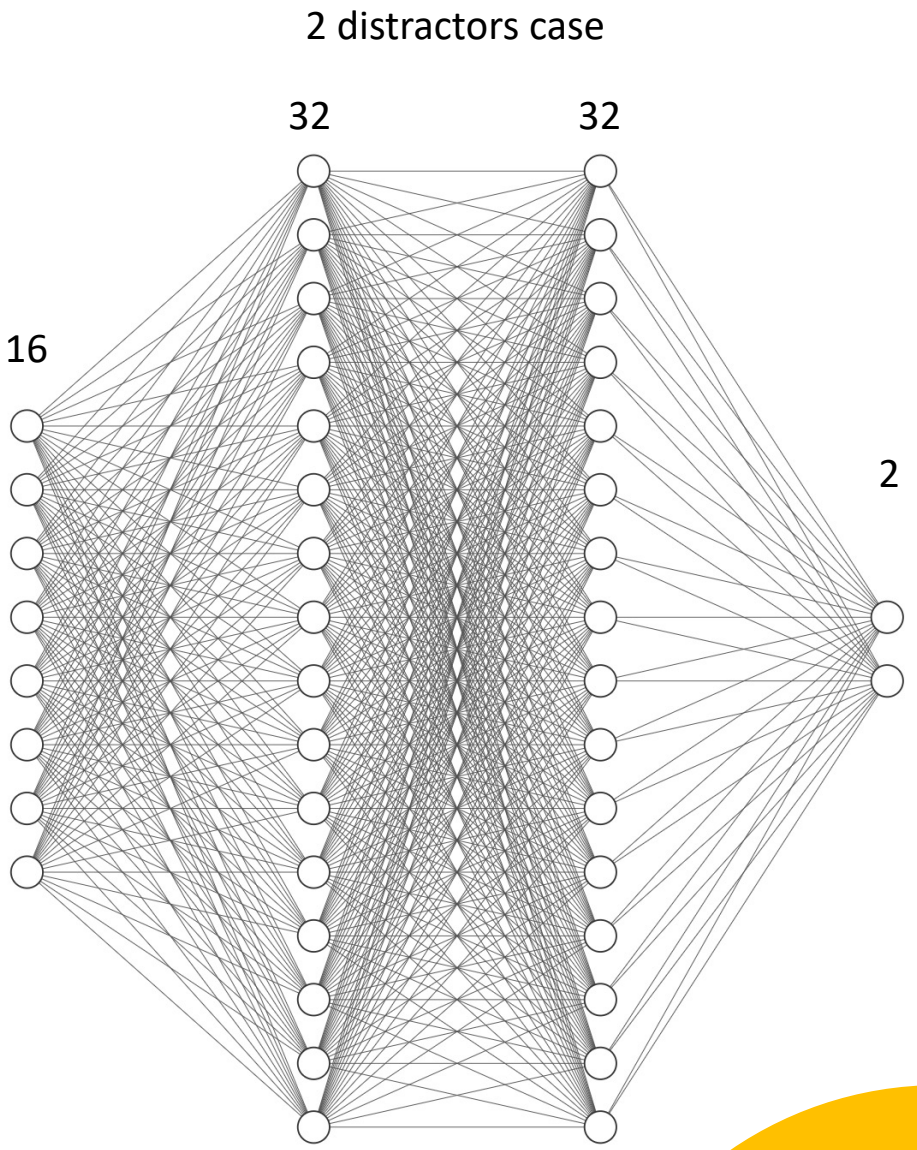
Baselines: Others

- **CNN** uses a 3-layer ConvNet with 16 kernels of size 3*3 and stride = 2 in each layer. ReLU is used after each Conv layer. Then the output is flattened and passed to PPO.



Baselines: Others

- **Oracle** uses the *states* (i.e. the (x, y, v_x, v_y) quadruple) of the target, agent, and distractor(s) rather than *graphical observations* as input.
- The states are directly passed into the policy network (actor) and value network (critic) in PPO.
- It is expected to have the best performance

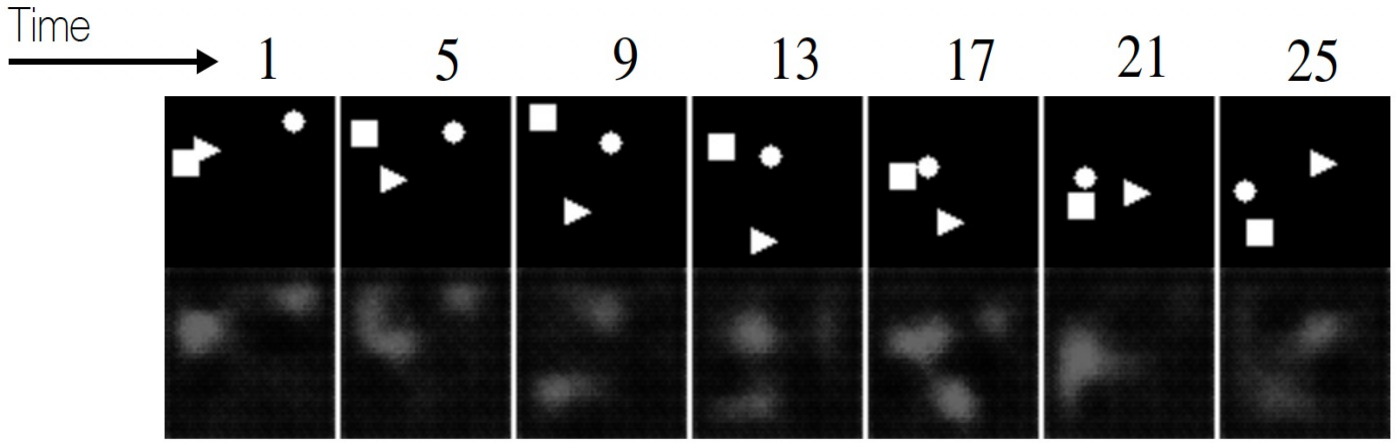


Outline

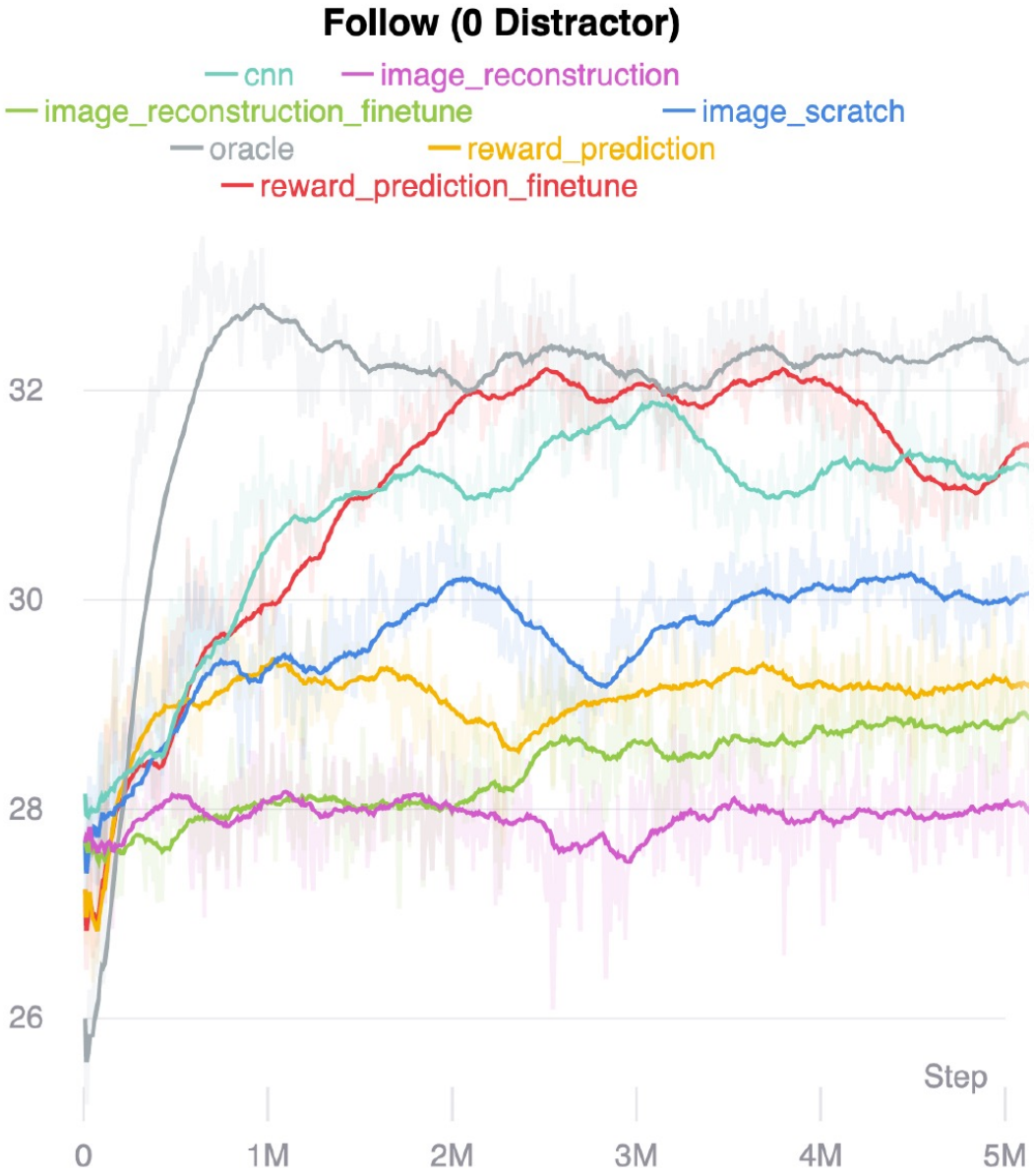
- Overview
- Experiment Details
- Results & Analysis
- Reflections & Implications

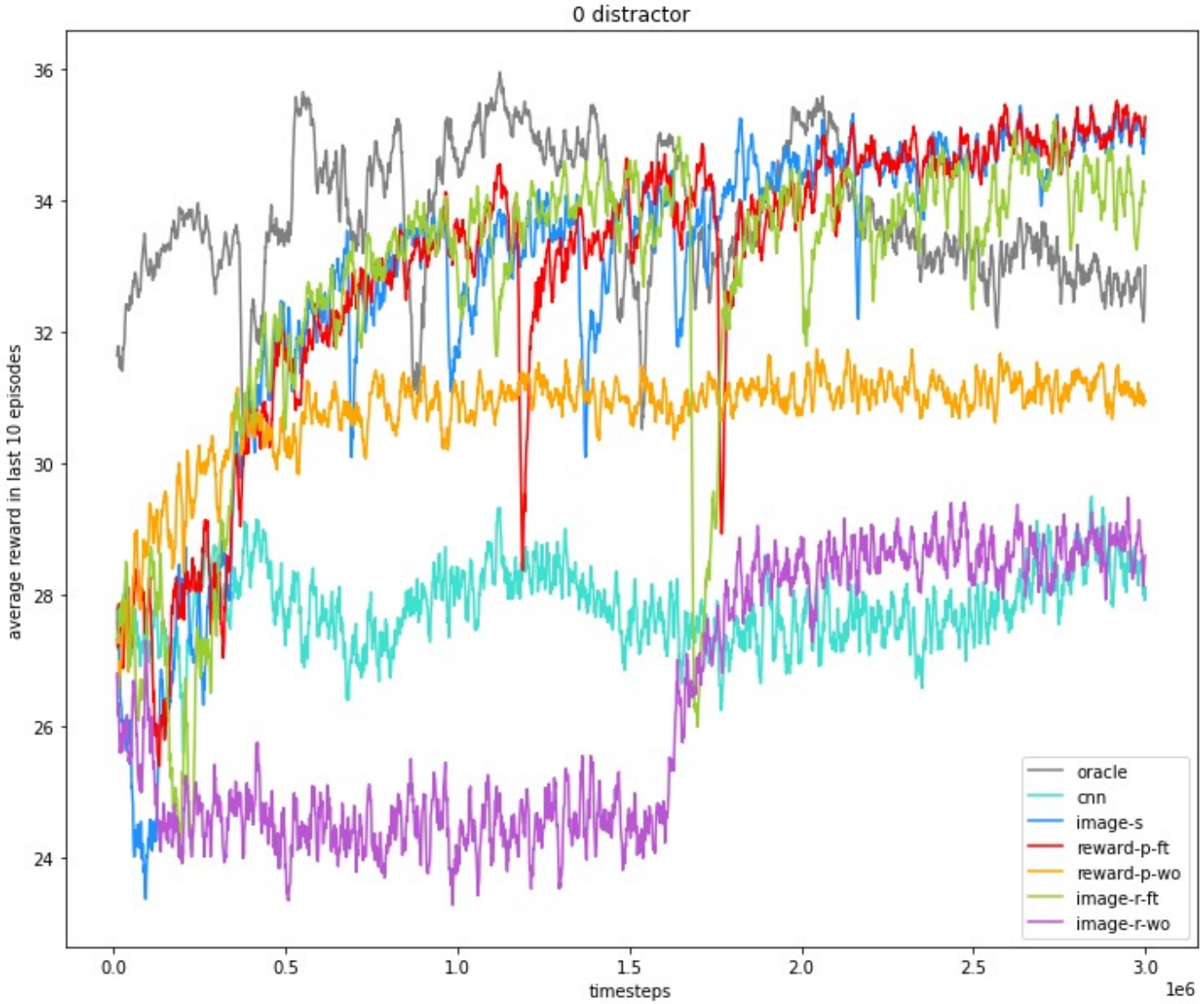
Does the reward-induced encoder capture useful information?

- I append a decoder (made of transposed Conv layers) to the reward-induced encoder and train them with image reconstruction loss (pixel-wise L2). Note that the parameters of the encoder should be fixed.

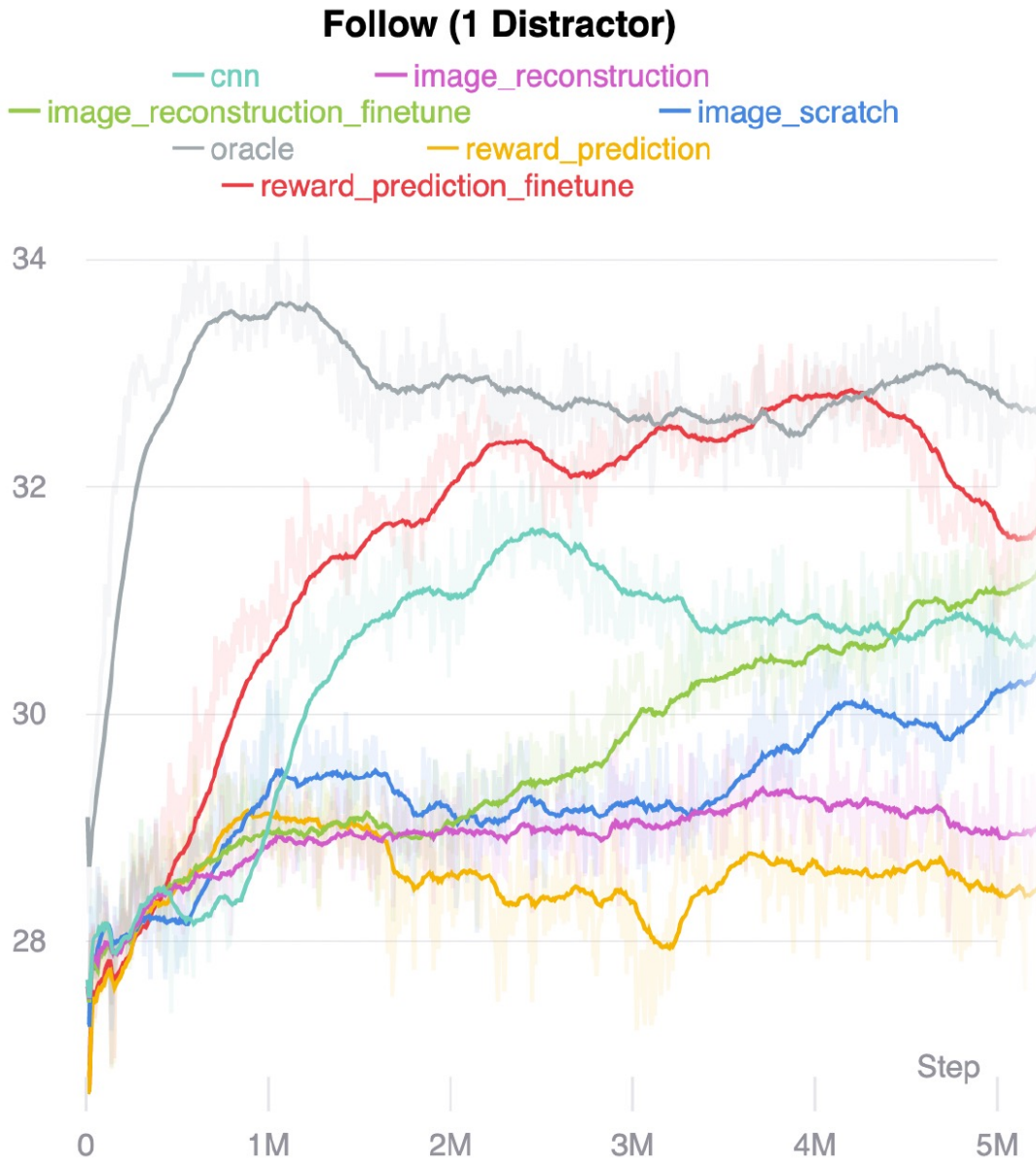


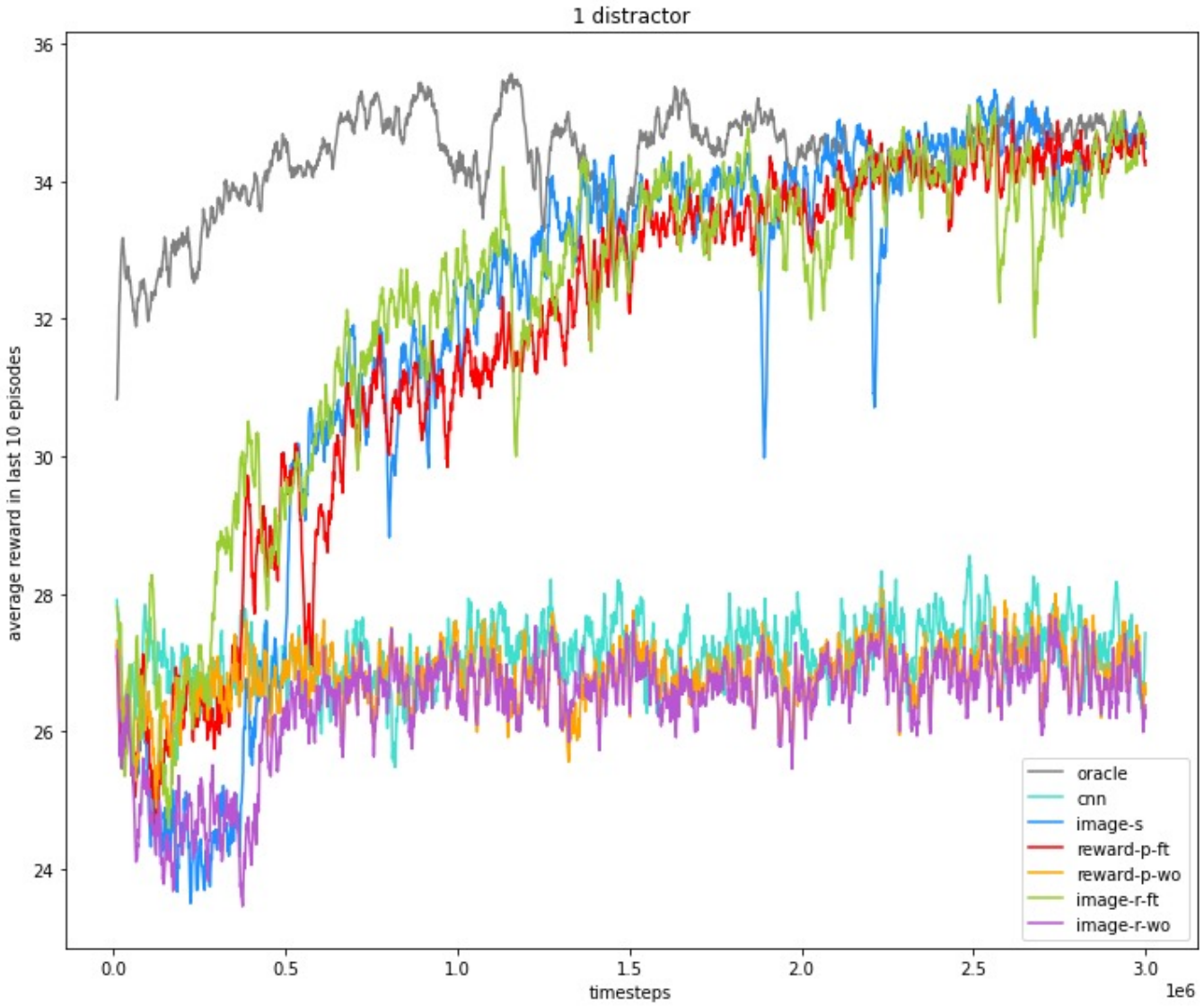
Reward Curves: 0 Distractor



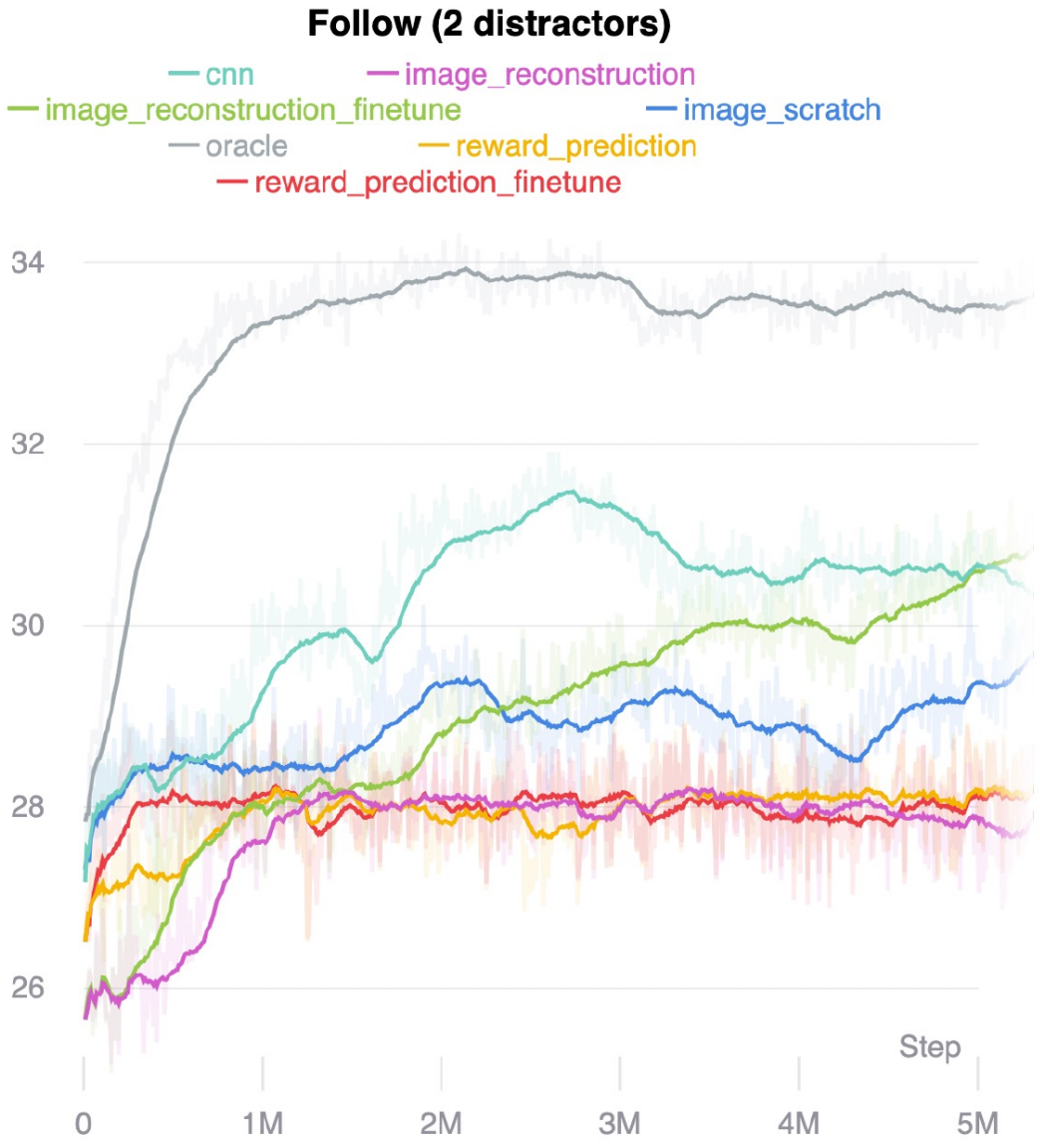


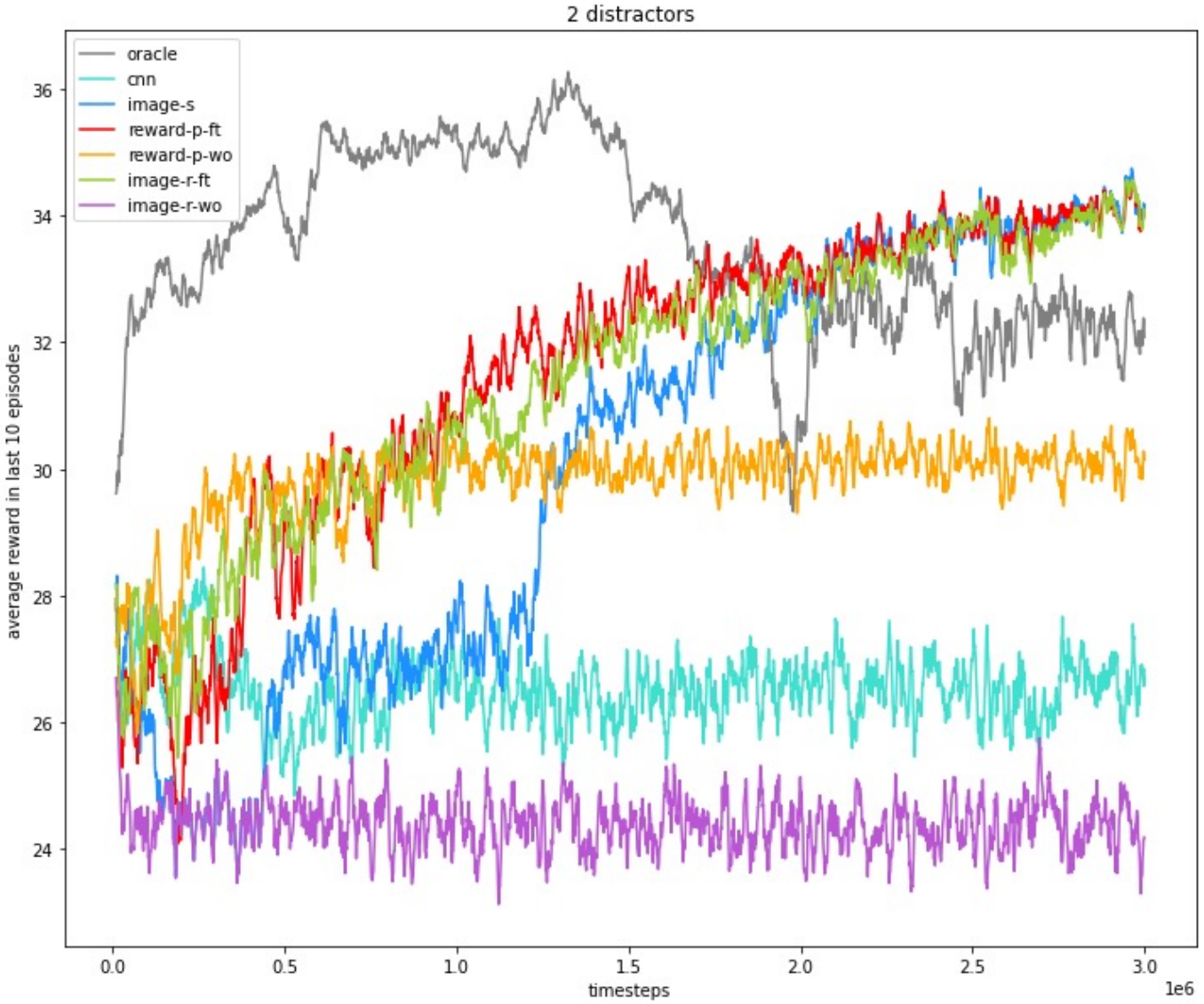
Reward Curves: 1 Distractor





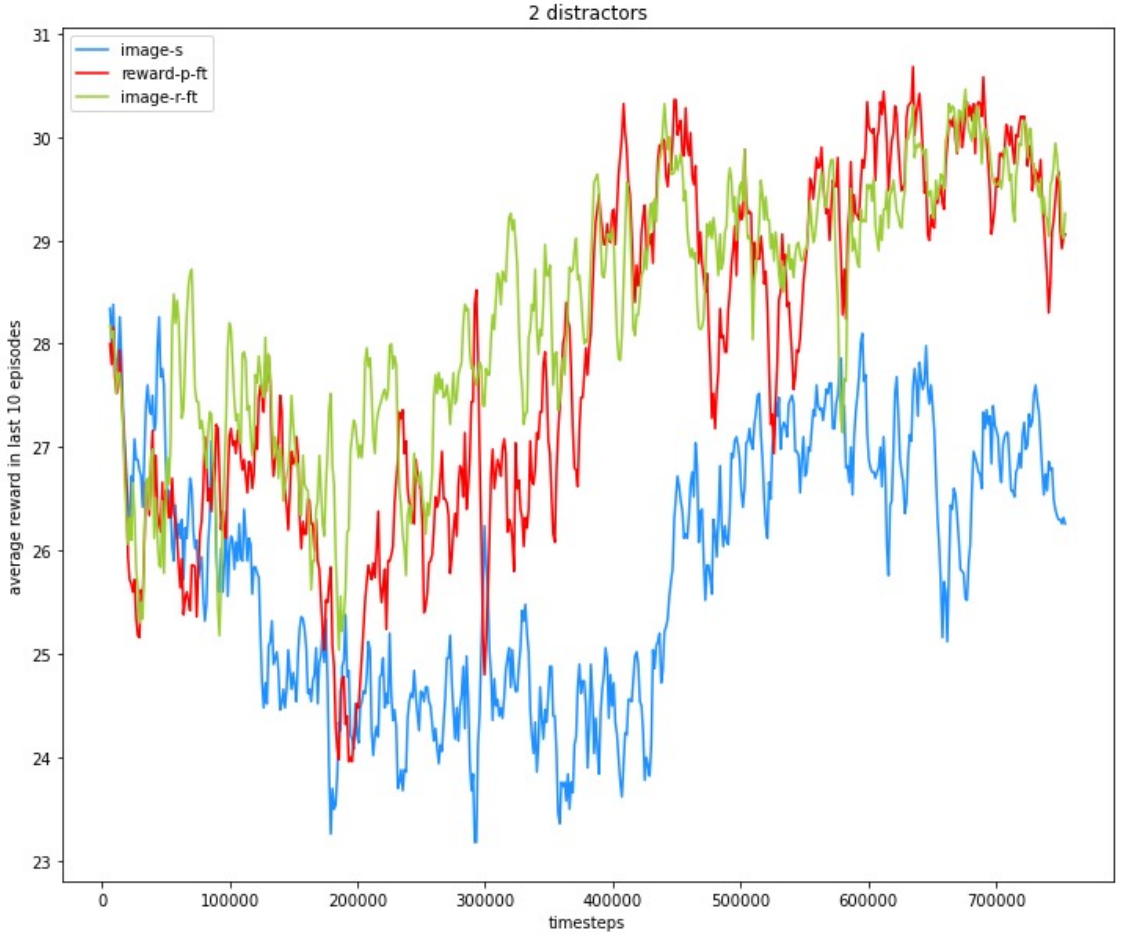
Reward Curves: 2 Distractors

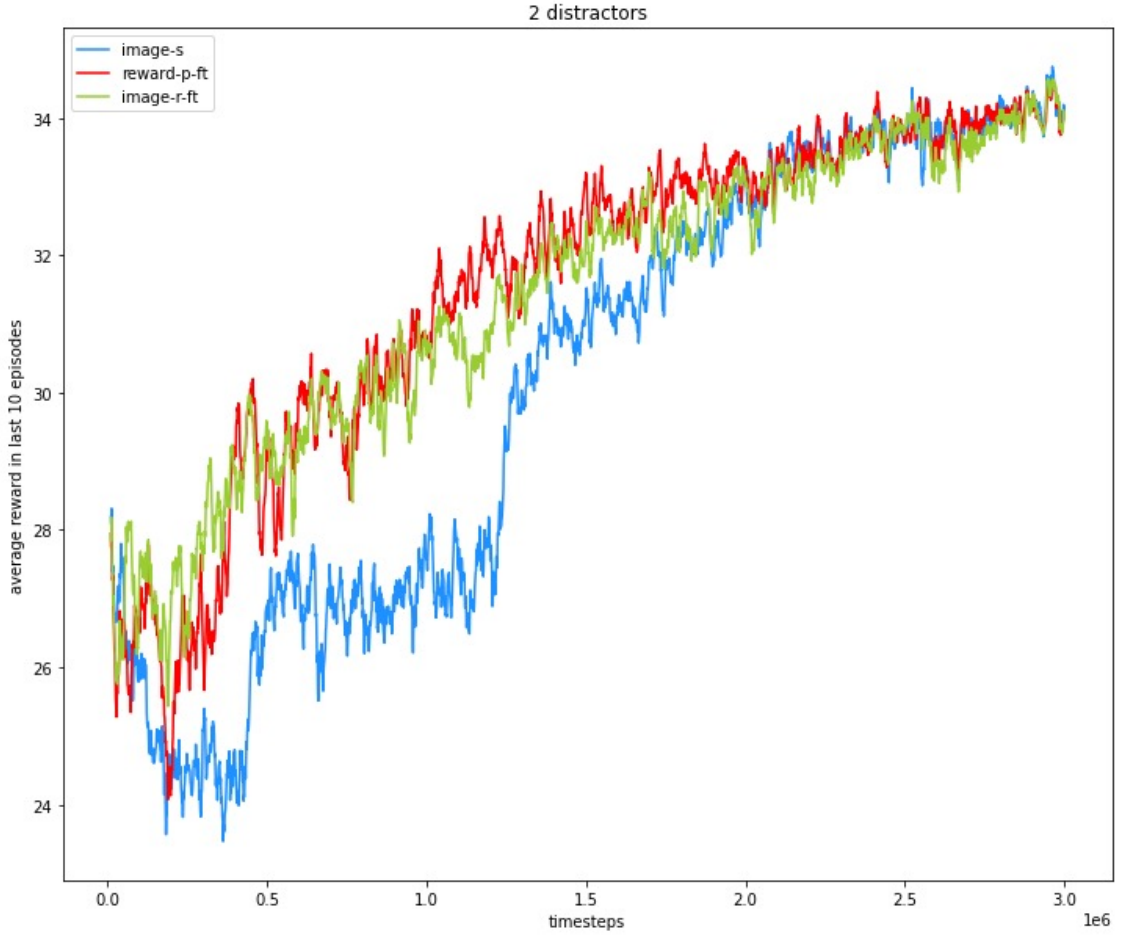




Initialization: Pretrain or NOT?

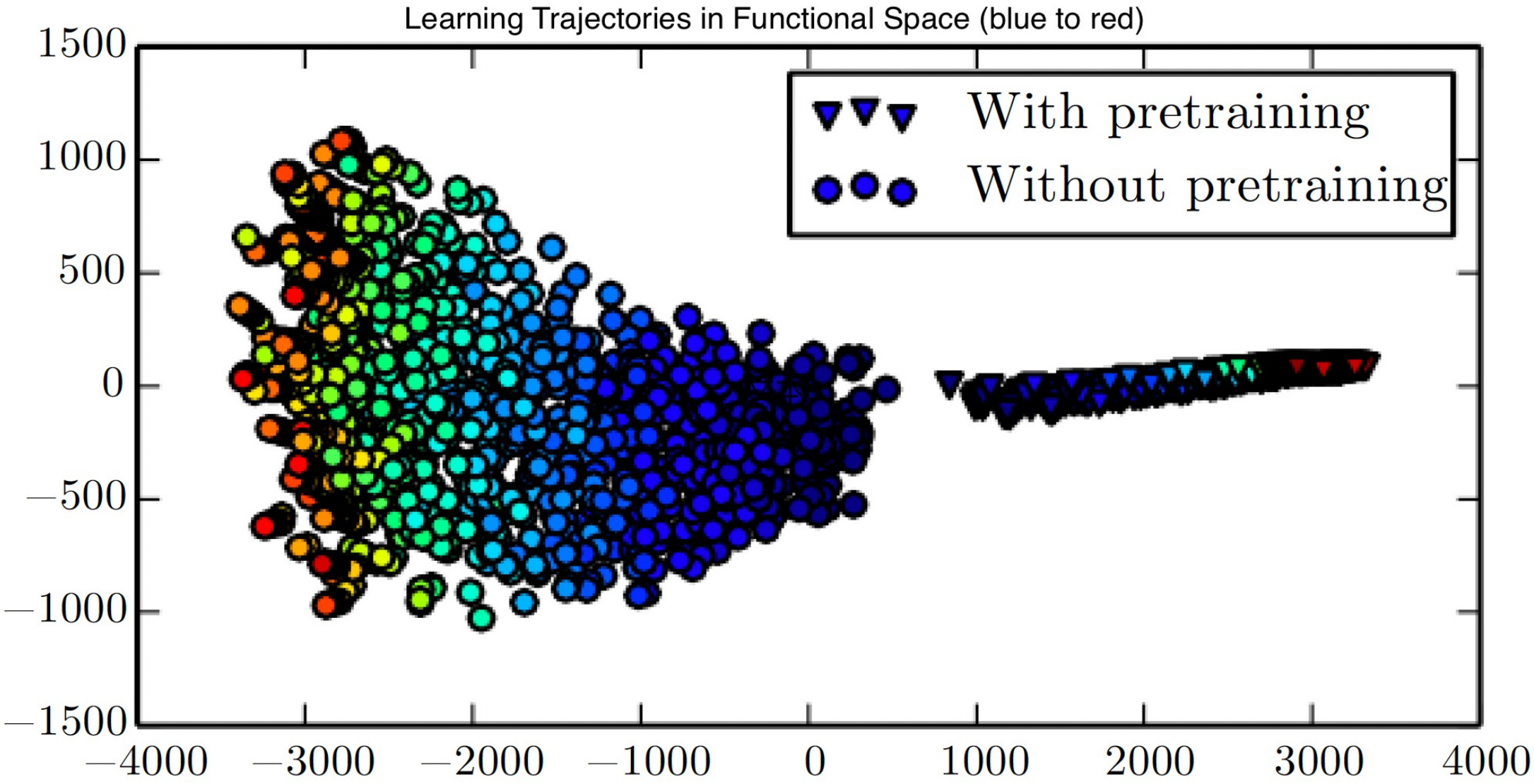
- Good initialization puts you at a better starting point (closer to the desired point)
- When finetuning is allowed, *eventually* a relatively poor initialization can achieve the same performance as the one with a better initialization





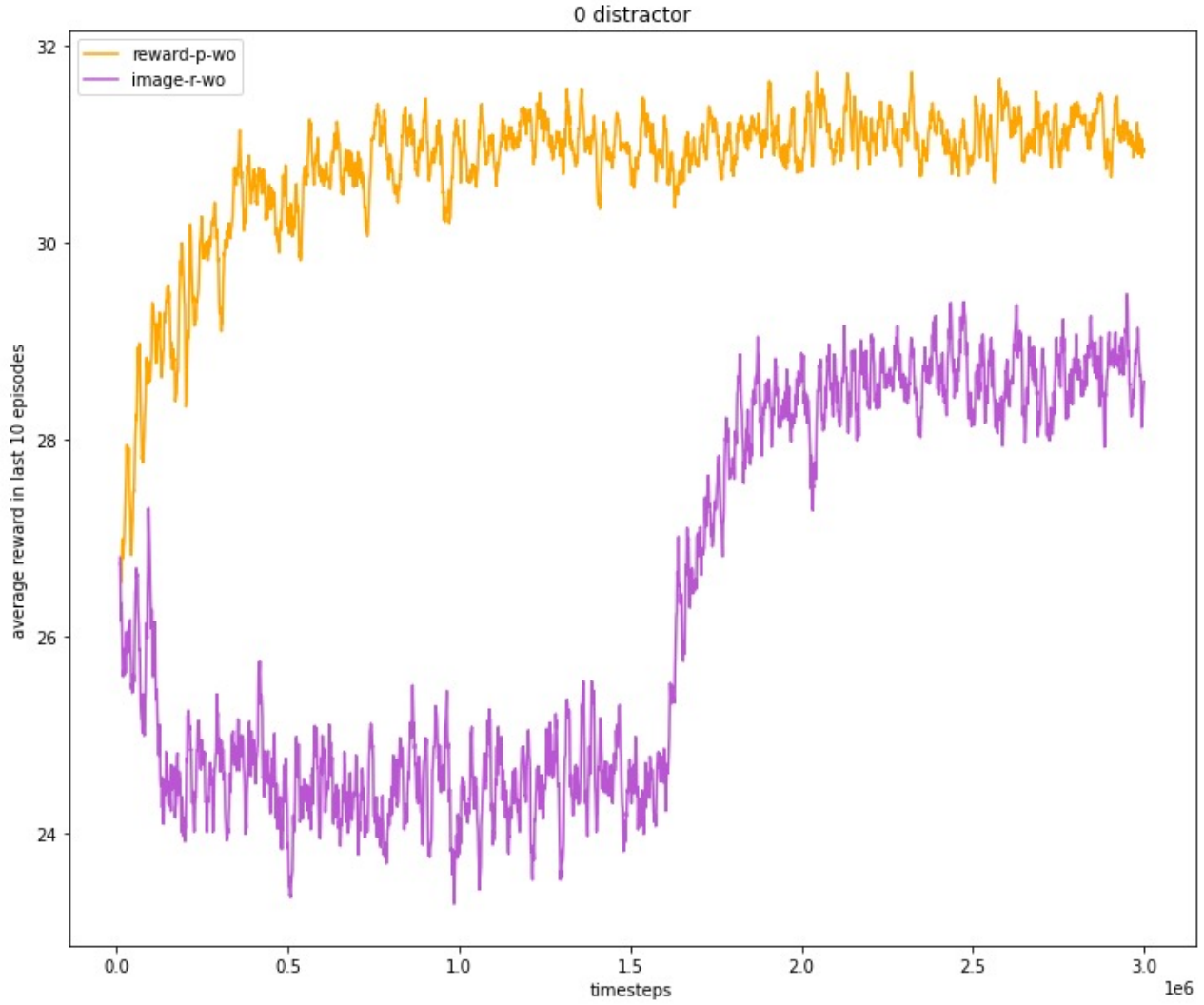
Theoretically, …

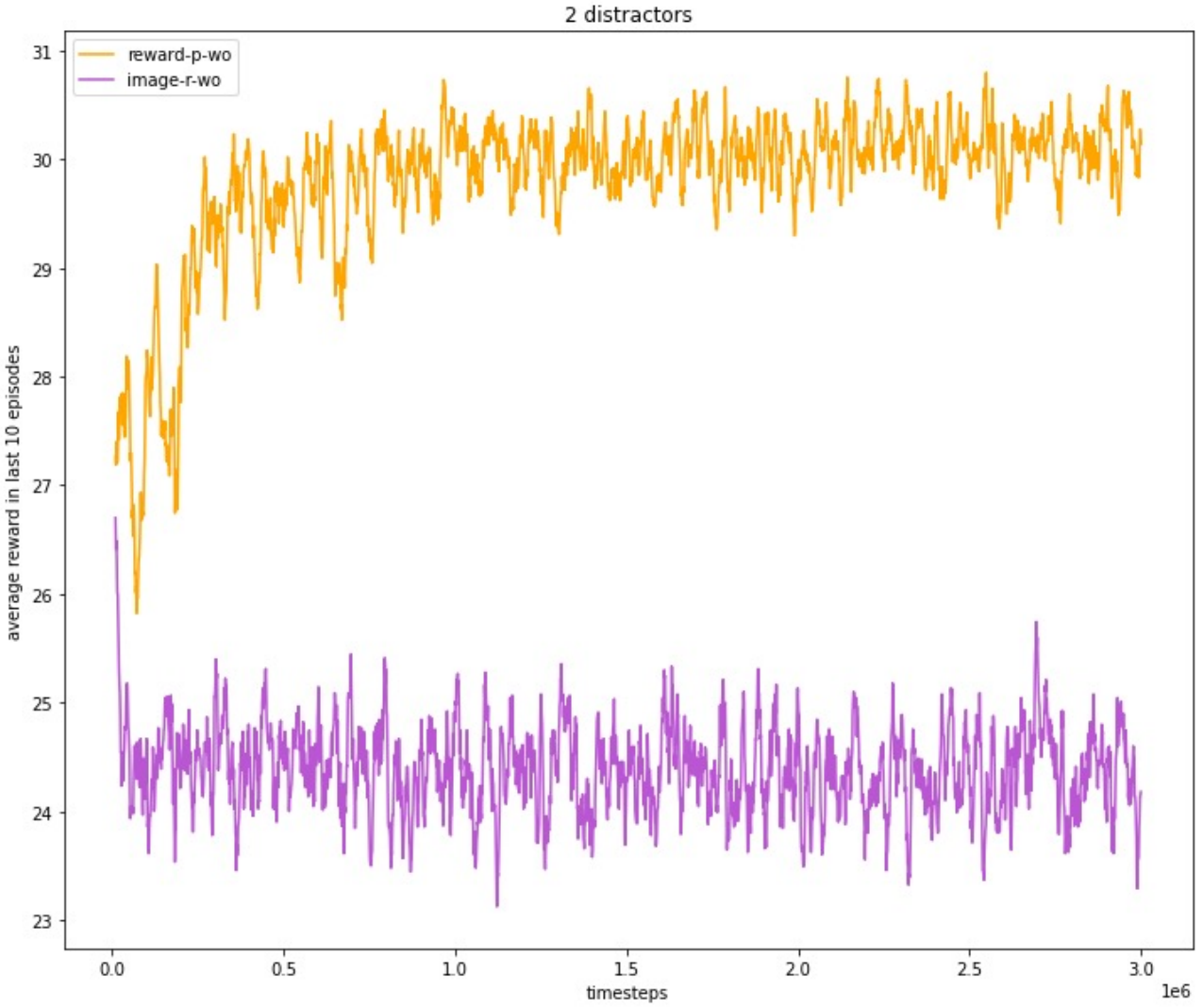
- “Training from random initialization is surprisingly robust.” [2]



Types of Representations

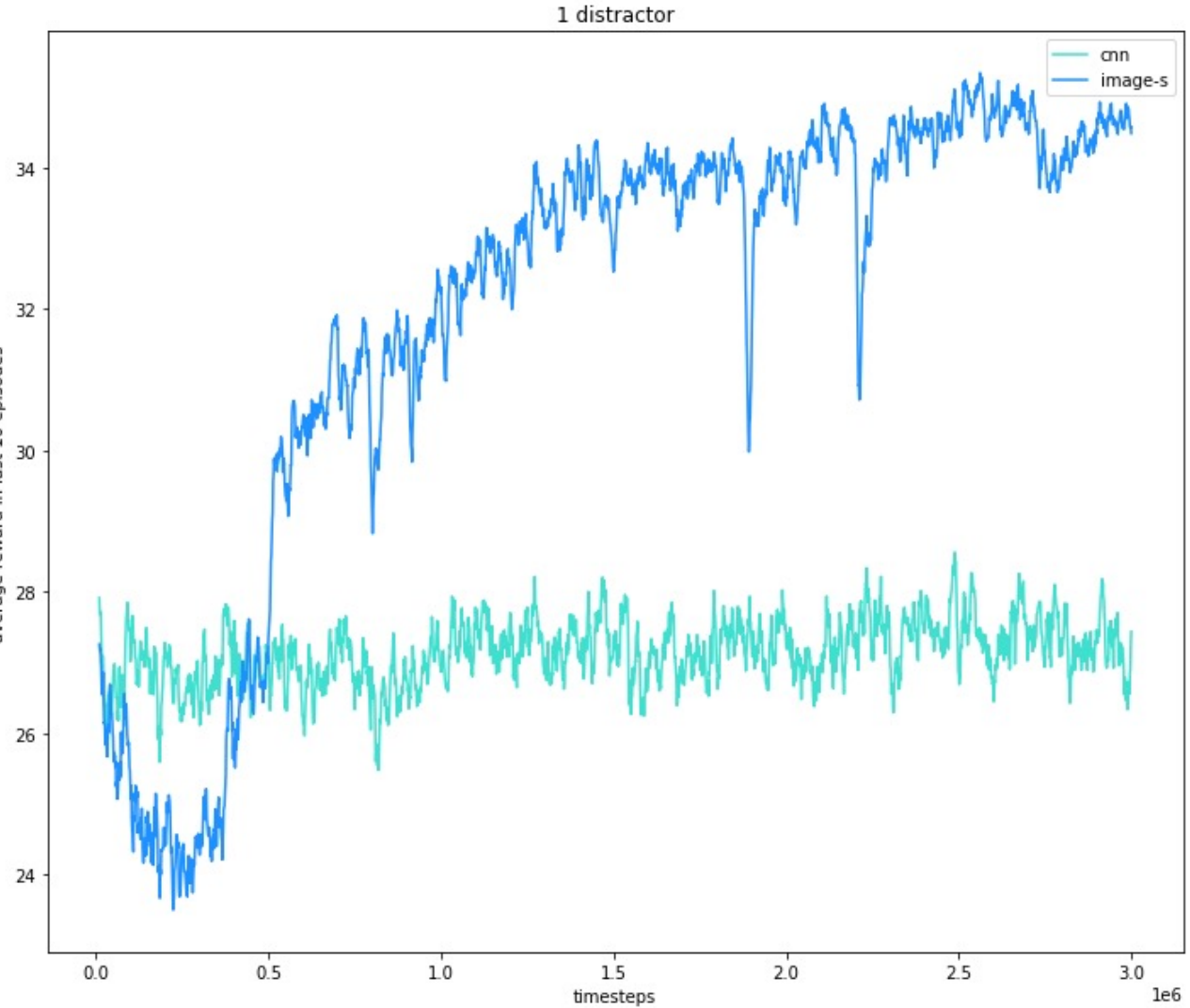
- Reward-induced encoder VS Image reconstruction encoder

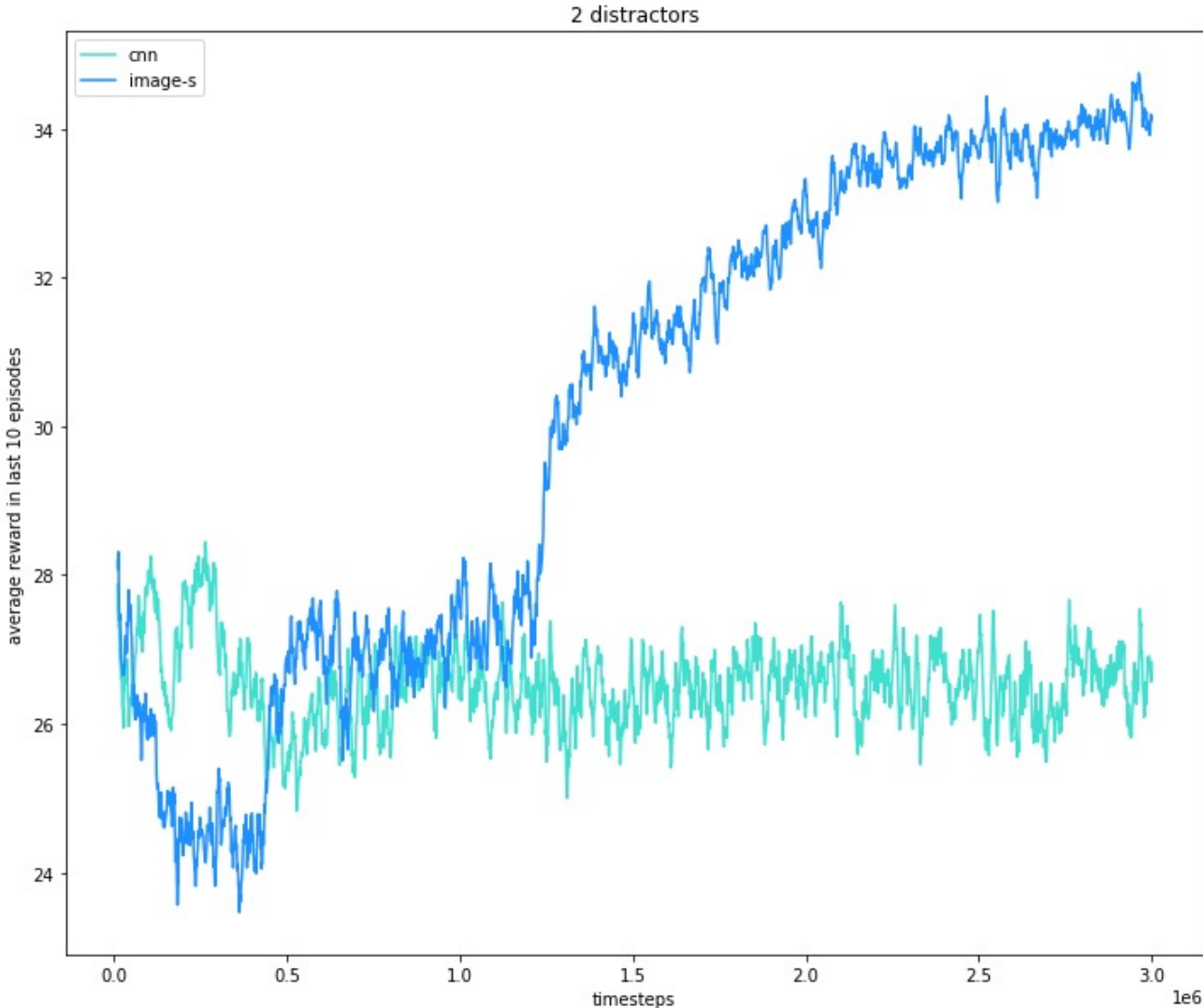




Representational Capacity of a Model

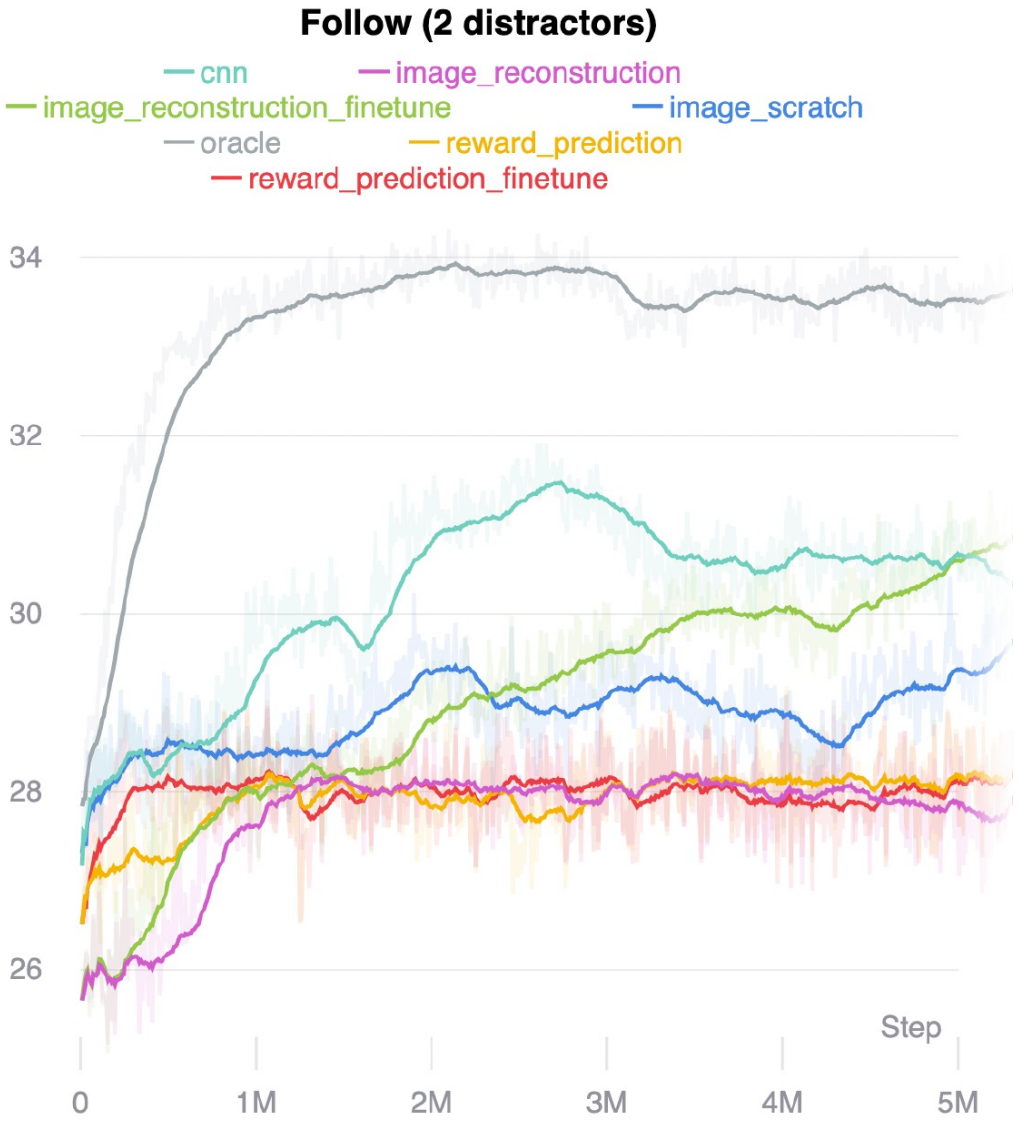
- CNN (3 Conv layers) VS Image-s (Encoder: 6 Conv layers with *BN* + 1 MLP layer)





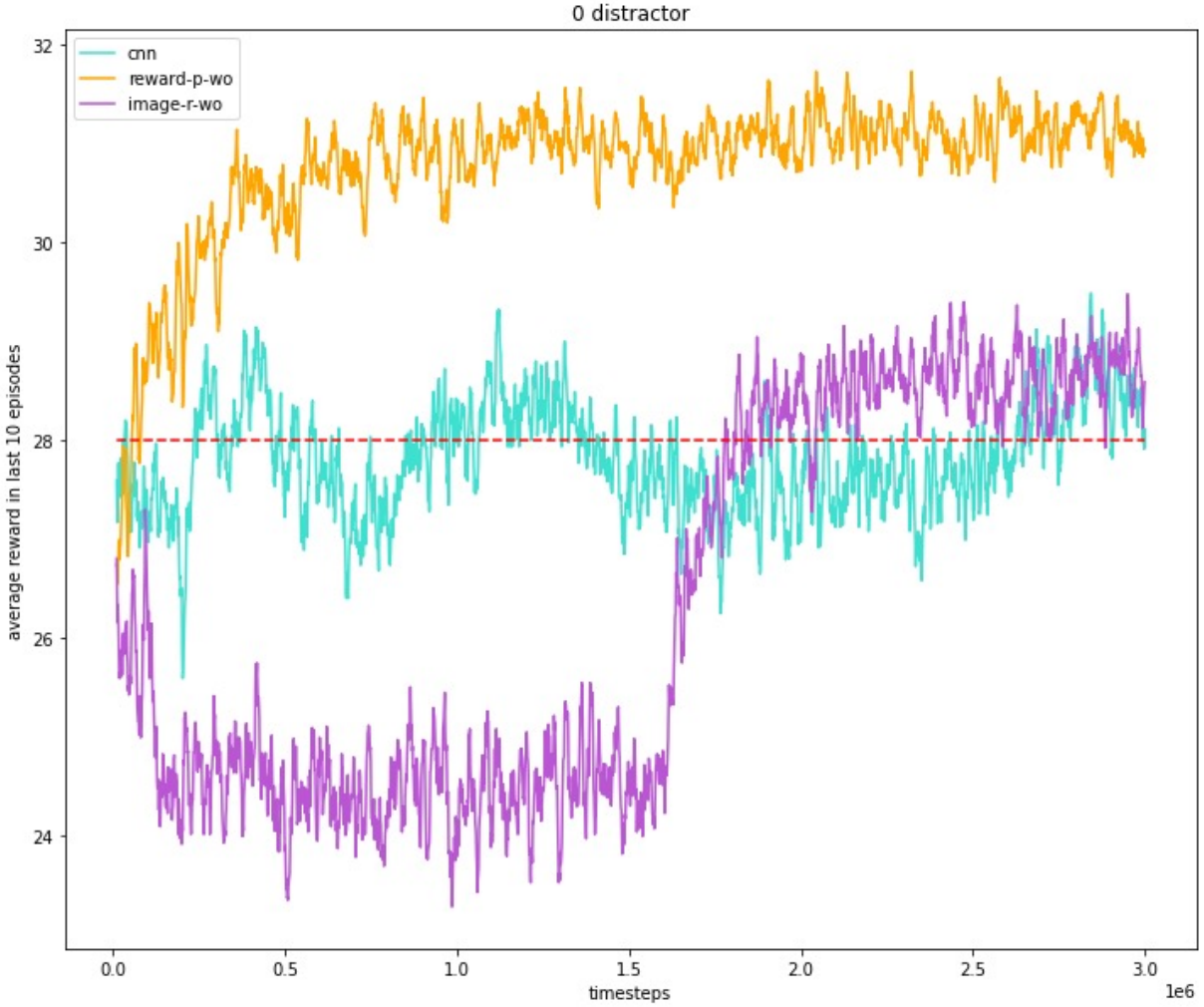
Opposite Conclusion?

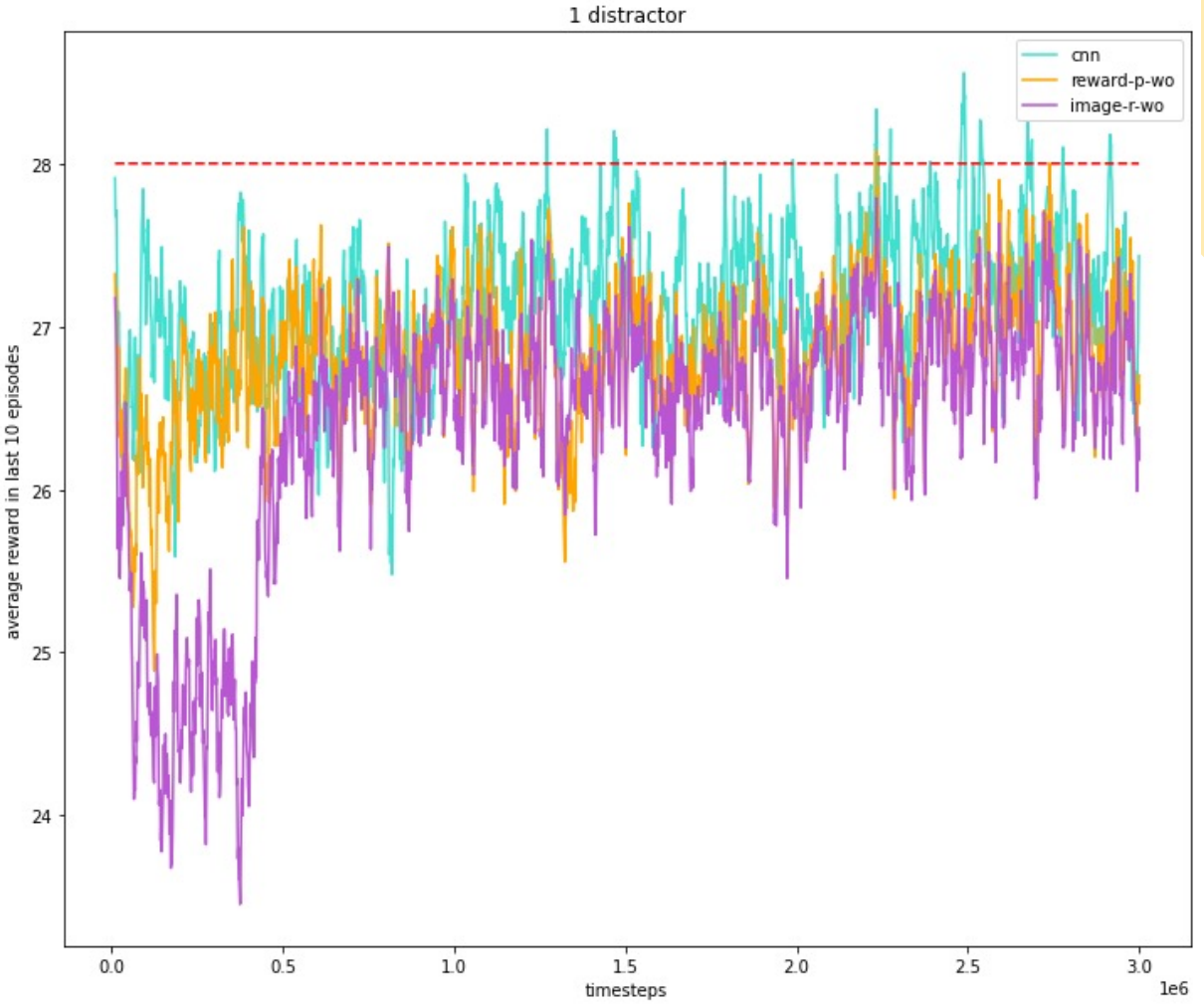
- “Note that *cnn* performs better than *image_scratch* because of its small sized CNN network suitable for RL.”[1]



Number of Distractors

- Having more distractors limits the performance of the “weaker” baseline models



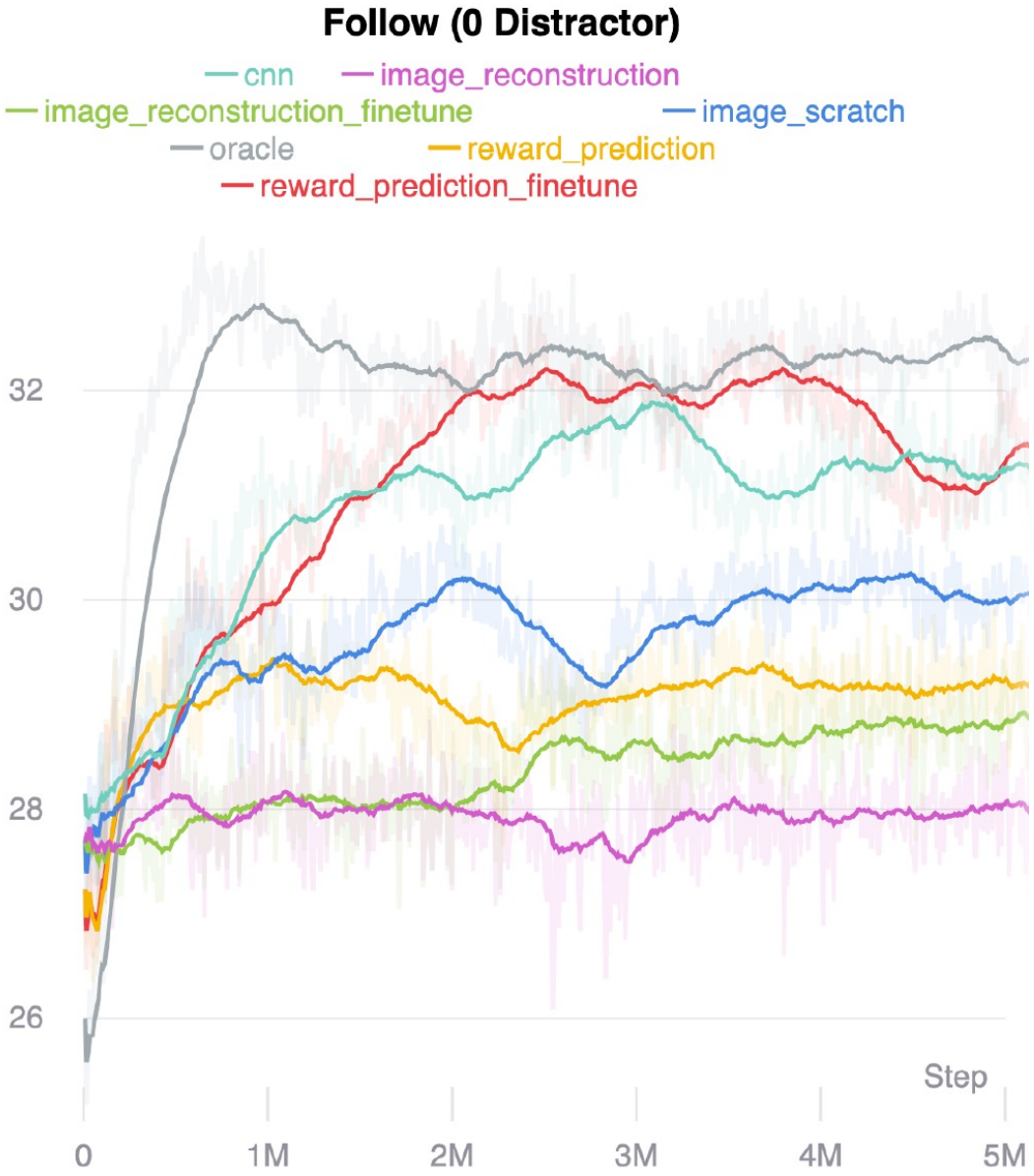


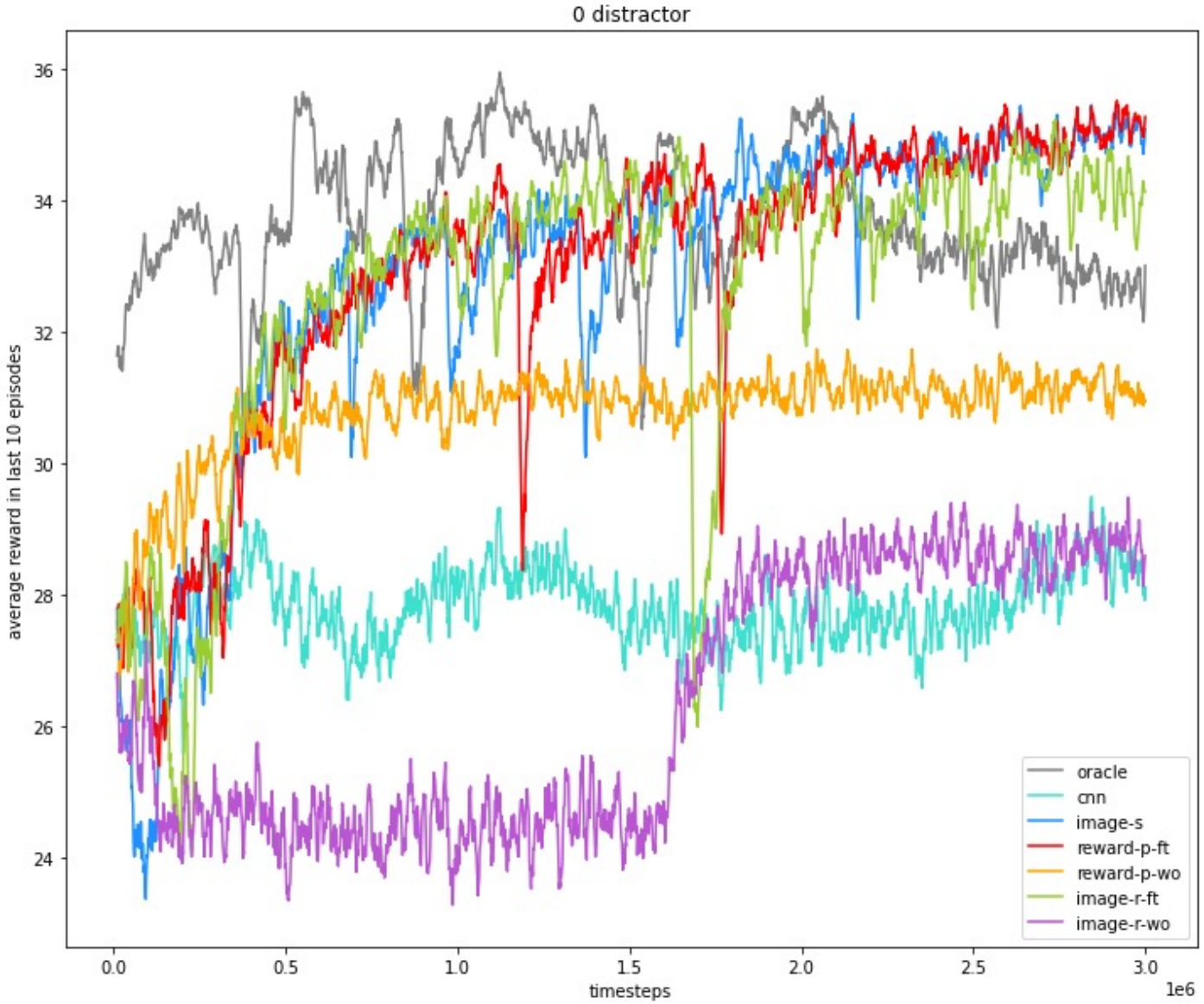
Abnormalities

- **Oracle**: in the *2 distractors* case, oracle's performance degrades after about 1.5M steps, eventually surpassed by those 3 with finetunable encoders.
- The 3 "strong" baseline models – **reward-p-ft**, **image-r-ft**, **image-s**:
  - Performances are highly aligned in my experiment, but NOT in the original* experiment
  - In my experiment, reward-p-ft is robust even in the *2 distractors* case, but it degrades in the original version
  - Sudden drops at some timesteps
- **Reward-p-wo**: In my experiment, it performs better in the *2 distractors* case than in the *1 distractor* case…
- **CNN**: seems to have good performance in the *original* experiment, but NOT in mine.
- My reward curves have higher variance than the original ones
- A wider range of rewards

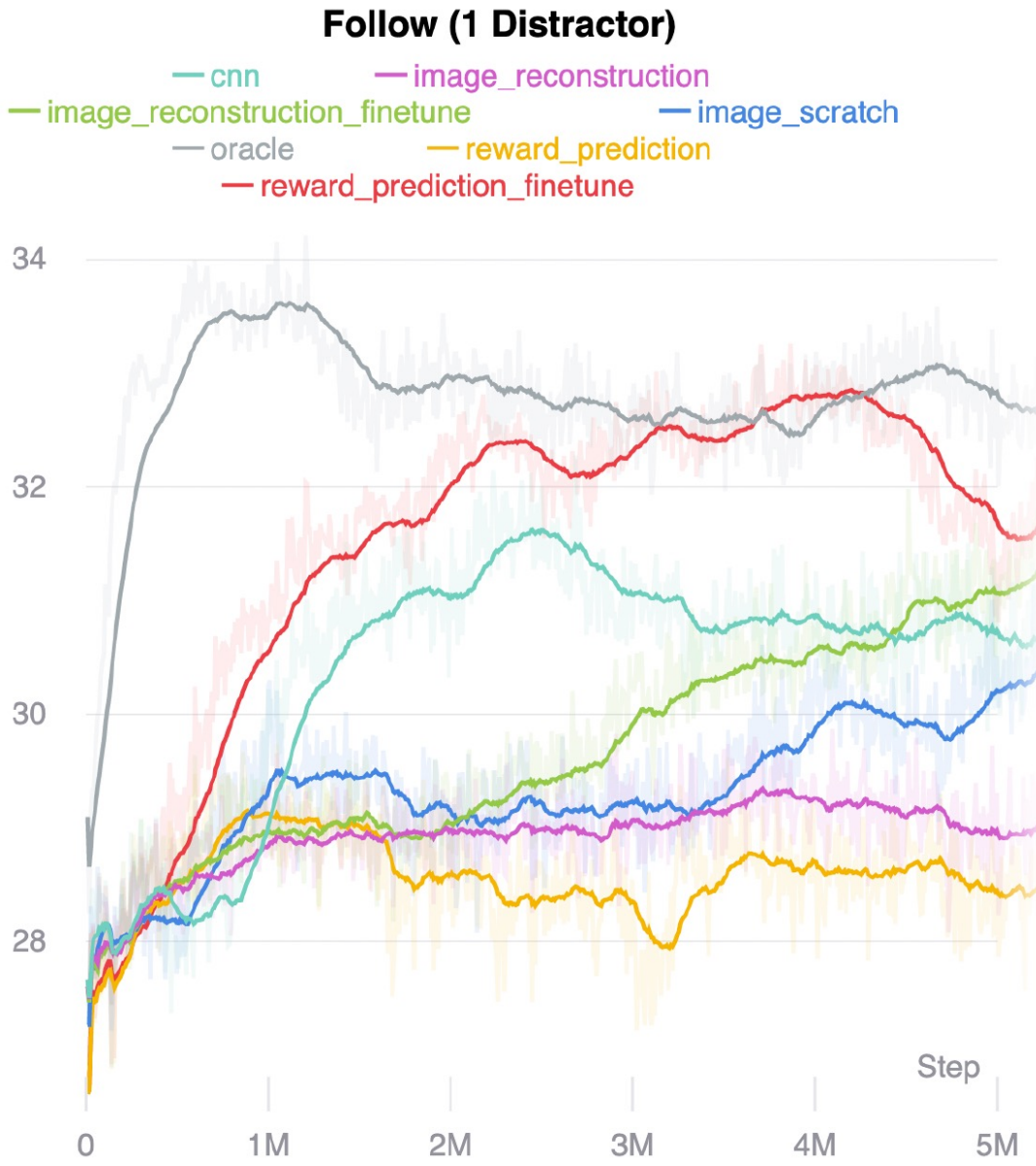
* Original means the experiments in the paper

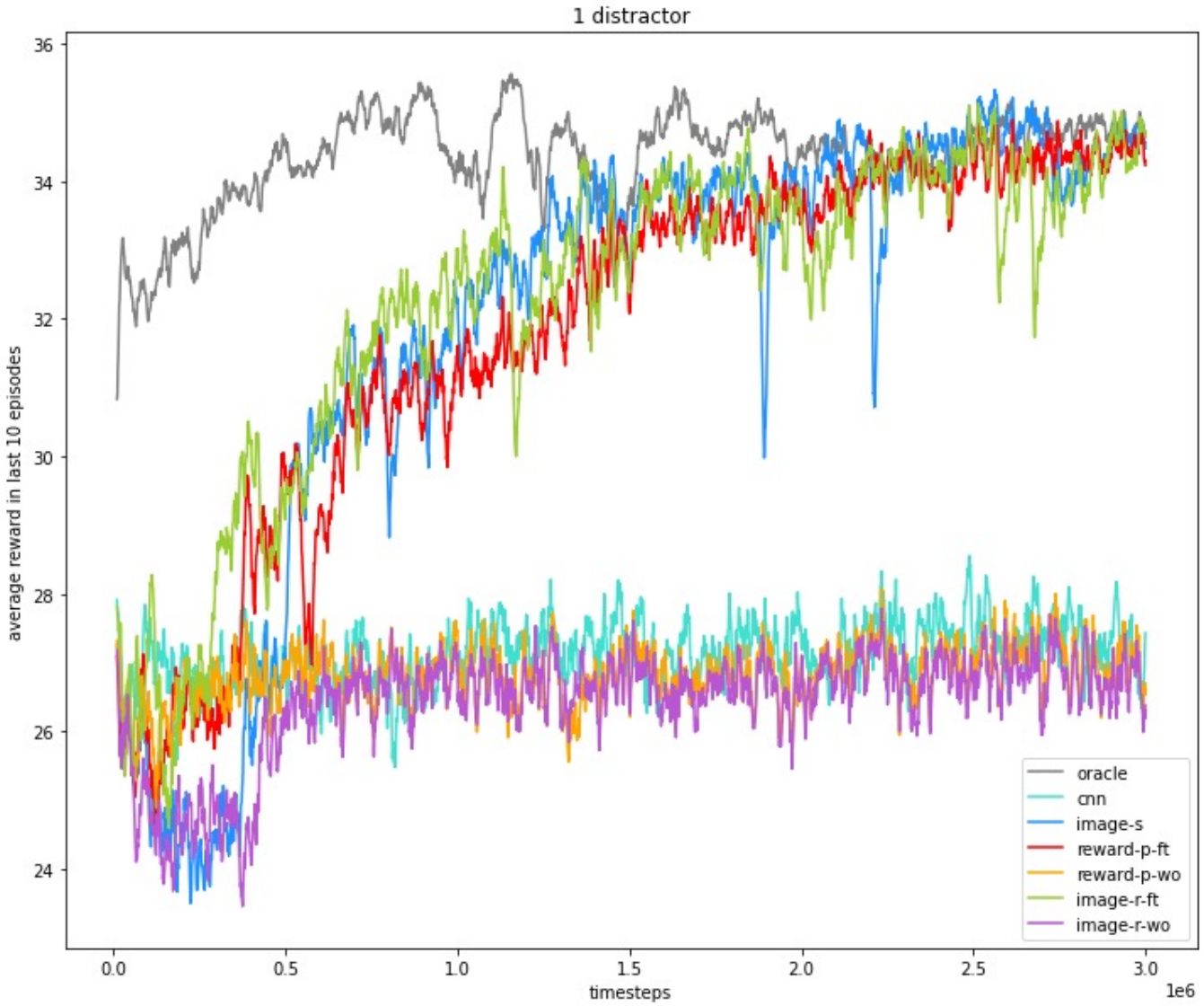
Reward Curves: 0 Distractor



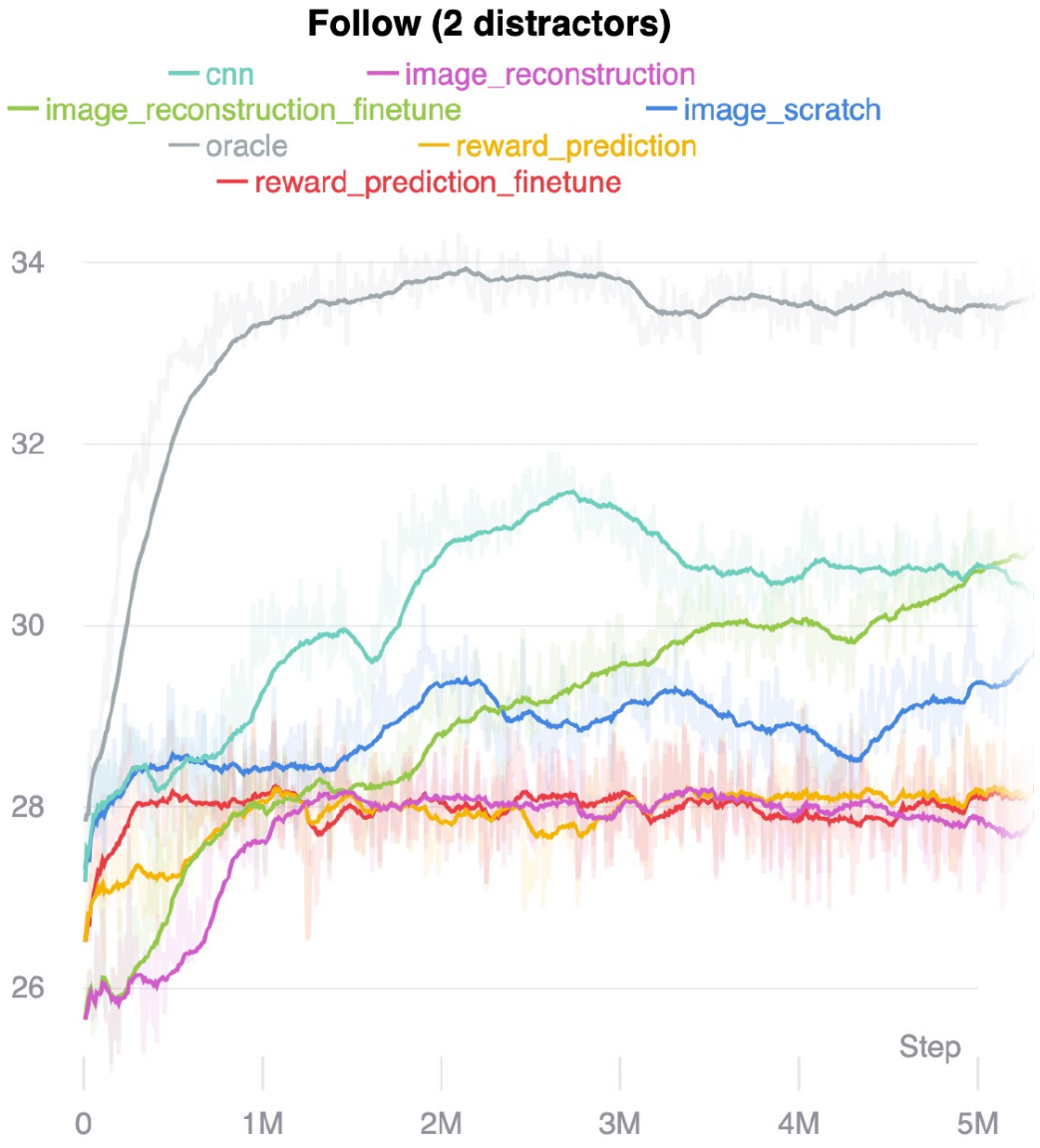


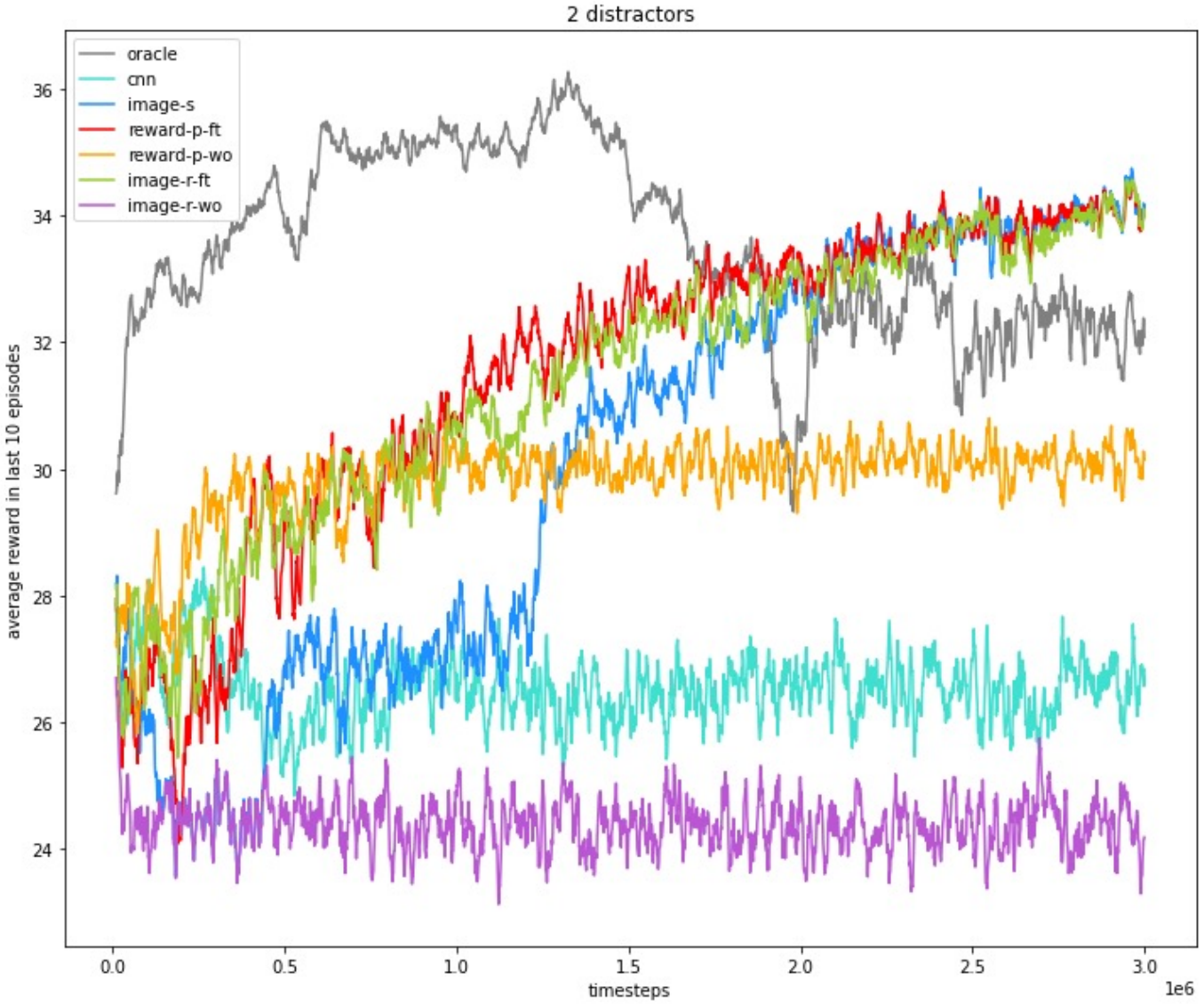
Reward Curves: 1 Distractor





Reward Curves: 2 Distractors





Outline

- Overview
- Experiment Details
- Results & Analysis
- Reflections & Implications

Variance Reduction

- Try other RL algorithms: DDPG/SAC
- General Advantage Estimation

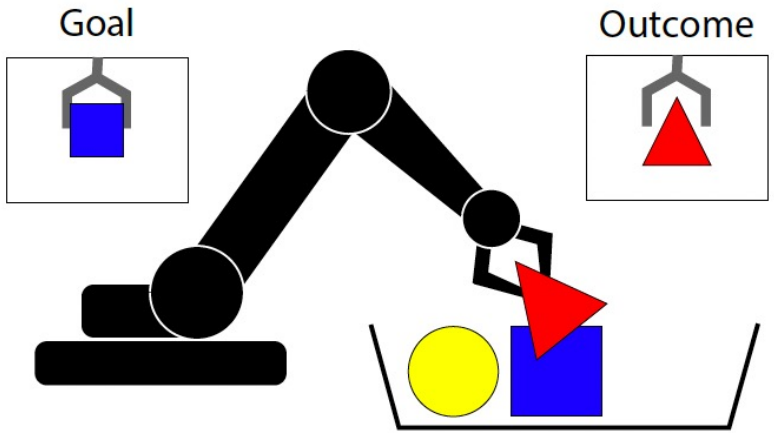
Representation Design

Grasp2Vec [4]:

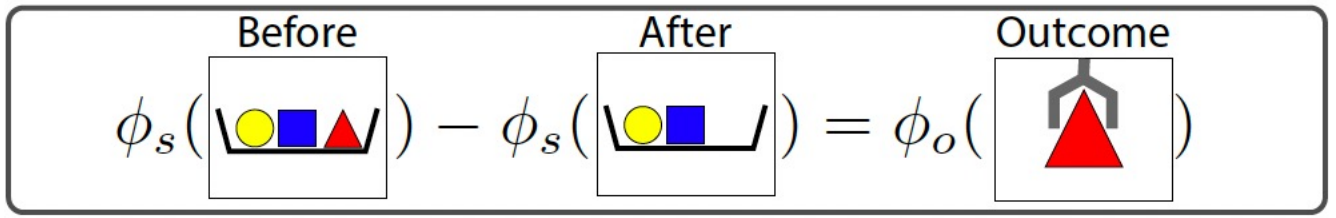
- Scenario: We want the robots to pick up an object queried by users among other objects
- Motivation: acquire effective object-centric representations of image inputs WITHOUT human labels of the identity of the object
- Idea: “object persistence” assumption for the embedding space $\phi(\mathcal{X})$

  Self-supervised learning to automatically improve

Instance Grasping



Representation Learning

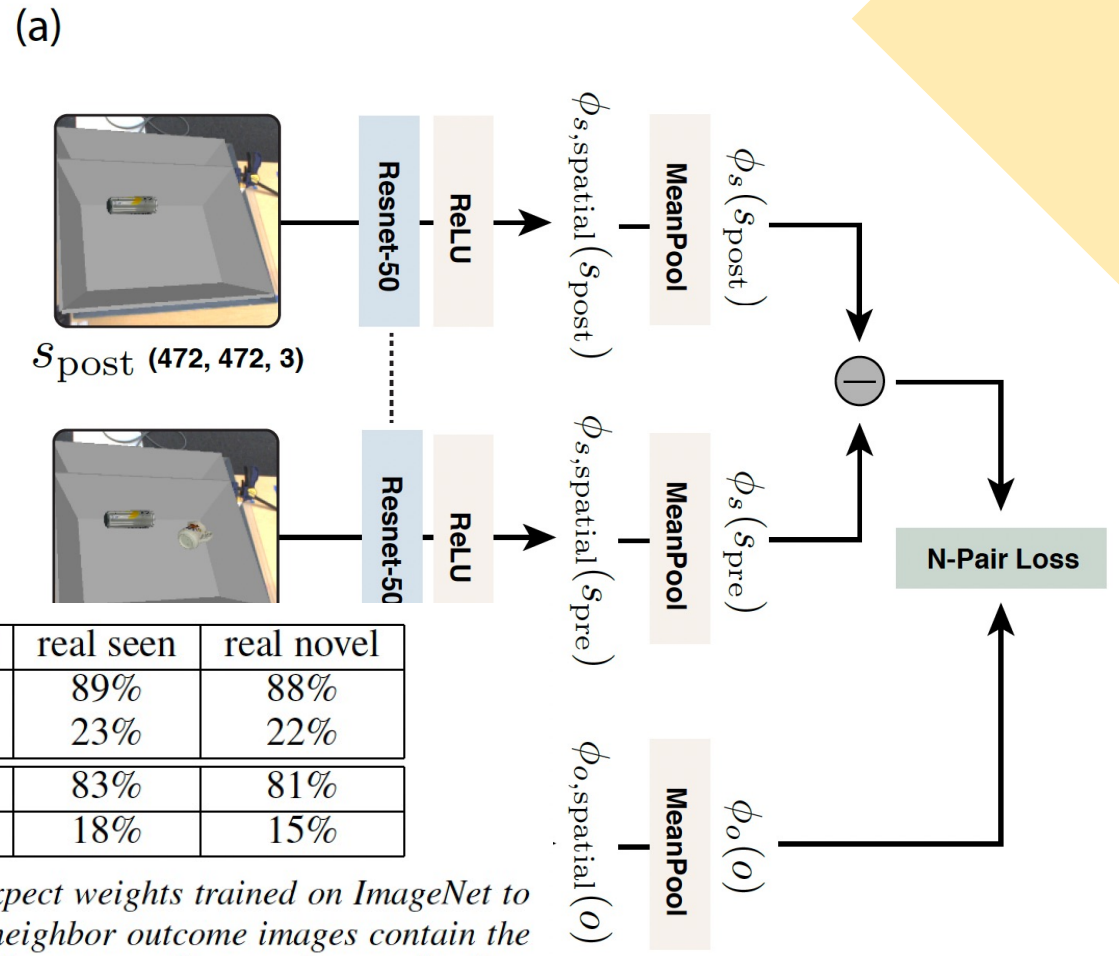


Representation Design

- (s_{pre}, s_{post}, o) -- the input image triple
- ϕ_s, ϕ_o -- embedding functions for states/object
- We want to impose $\phi_s(s_{pre}) - \phi_s(s_{post}) \approx \phi_o(o)$

"positive"

"anchor"



- N-Pair Loss (self-supervised)

$$\text{NPairs}(a,p) = \sum_{i<B} -\log\left(\frac{e^{a_i \top p_i}}{\sum_{j<B, j \neq i} e^{a_i, p_j}}\right) + \lambda(||a_i||_2^2 + ||p_i||_2^2).$$

- Results

	sim seen	sim novel	real seen	real novel
Retrieval (ours)	88%	64%	89%	88%
Outcome Neighbor (ImageNet)	—	—	23%	22%
Localization (ours)	96%	77%	83%	81%
Localization (ImageNet)	—	—	18%	15%

Table 1: Quantitative study of Grasp2Vec embeddings. As we cannot expect weights trained on ImageNet to exhibit the retrieval property, we instead evaluate whether two nearest neighbor outcome images contain the same object. For object localization, the ImageNet baseline is performed the same as the grasp2vec evaluation. See Figure 4b for examples heatmaps and Appendix A for example retrievals.

What can we do with Representation Learning?

- Dimension Reduction
- Few-shot Learning
- Transfer Learning: domain/task adaption
- NLP pre-trained language models – BERT, GPT

References

[1] Ayush Jain, Youngwoon Lee, and Karl Pertsch. "Reward-Induced Representation Learning". In: ()

[2] Kaiming He, Ross Girshick, and Piotr Dollár. "Rethinking imagenet pre-training". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4918–4927

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016

[4] Eric Jang et al. "Grasp2vec: Learning object representations from self-supervised grasping". In: *arXiv preprint arXiv:1811.06964* (2018)

[5] Michael Tschannen, Olivier Bachem, and Mario Lucic. "Recent advances in autoencoder-based representation learning". In: *arXiv preprint arXiv:1812.05069* (2018)

Thank you!

Special thanks to Jesse Zhang!