Reinforcement Learning with Unsupervised Auxiliary Tasks

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo,
David Silver & Koray Kavukcuoglu
Google DeepMind

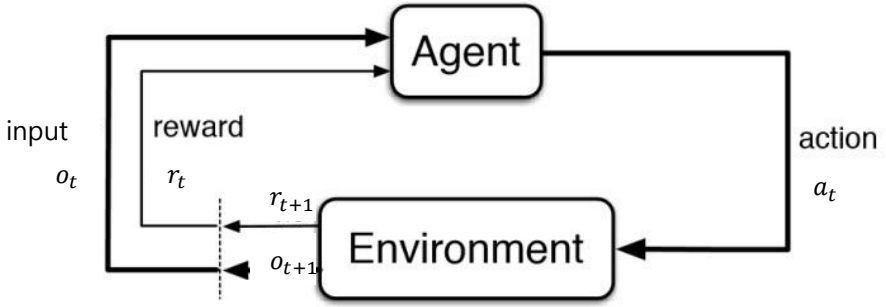
Nature 2016

Presented by Yoonsung Kim

INTRODUCTION

- Agent have an abundance of possible learning targets other than reward.
- Traditionally, unsupervised learning was typically used to accelerate the acquisition of a useful representation.
- This research used auxiliary objectives to obtain useful representations for long-term goals.

Definitions



- A : action set
- $o_t \in \mathbb{R}^d$: observation at time t
- a_t : action at time t
- r_t : reward at time t
- $s_t = f(o_1, r_1, a_1, x_2, \ldots, o_t, r_t)$: state
- $R_{t:t+n} = \sum_{i=1}^{n} \gamma^i r_{t+i}$: discounted return (γ : discount factor)
- $V^{\pi}(s) = \mathbb{E}[R_t | s_t = s, \pi]$: value function
- $Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$: action-value function

A3C Framework

$$\mathcal{L}_{\text{A3C}} \approx \mathcal{L}_{\text{VR}} + \mathcal{L}_{\pi} - \mathbb{E}_{s\sim\pi}\left[\alpha H(\pi(s,\cdot,\theta)\right]$$

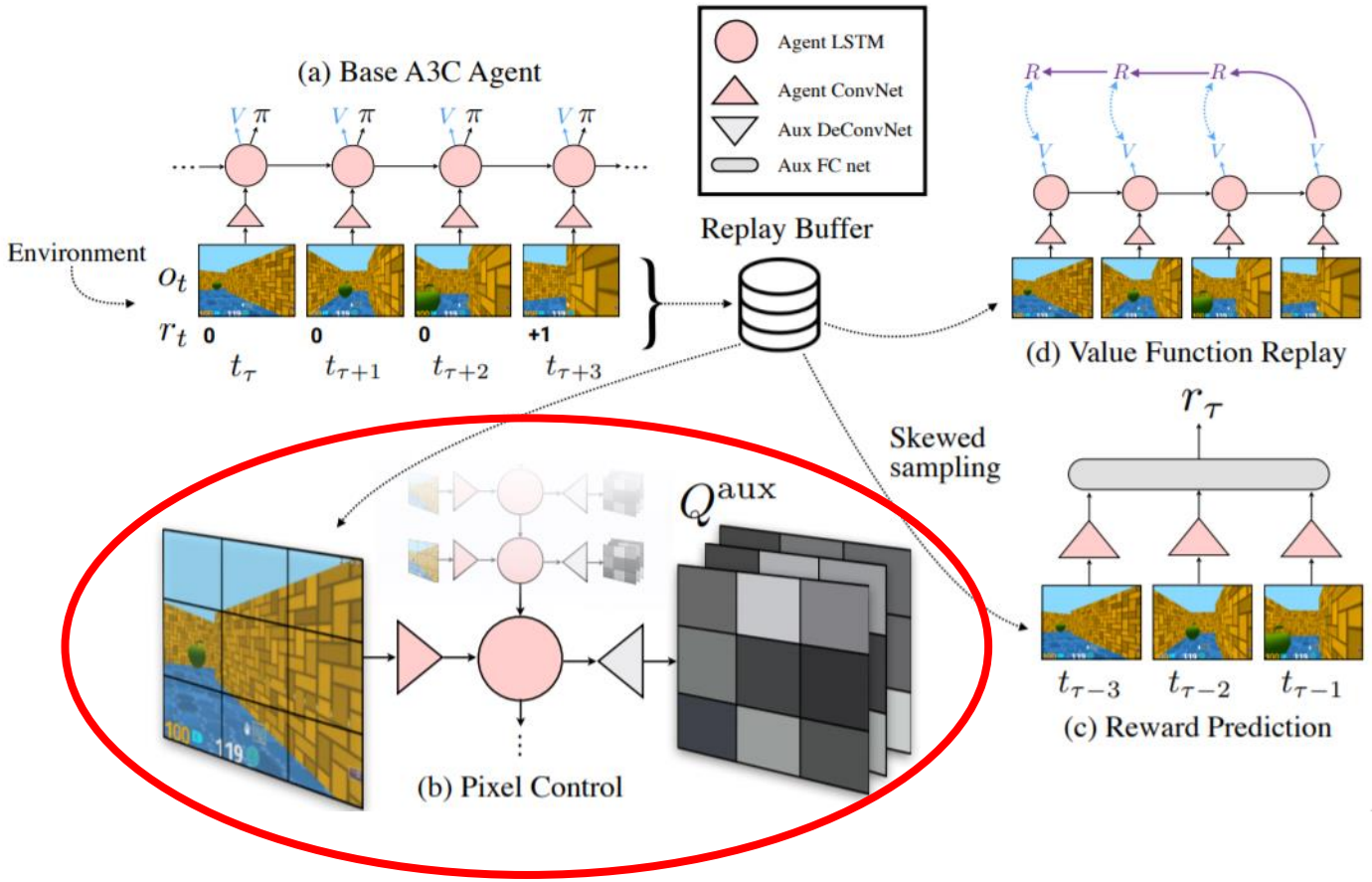
$$\mathcal{L}_{\text{VR}} = \mathbb{E}_{s\sim\pi}\left[\left(R_{t:t+n} + \gamma^{n} V(s_{t+n+1},\theta^{-}) - V(s_t,\theta)\right)^2\right]$$

Auxiliary Control Tasks

- Auxiliary control tasks : Additional pseudo-reward functions in the environment
- Auxiliary task $c \in C$ is defined by a reward function $r^{(c)}: S \times A \to \mathbb{R}$
  (S : space of possible states, A : space of available actions)
- Objective : to maximize total reward across all auxiliary tasks
  $argmax_\theta \mathbb{E}_\pi[R_{1:\infty}] + \sum_{c \in C} \lambda_c \mathbb{E}_{\pi_c}[R^{(c)}_{1:\infty}]$
  ($\pi^{(c)}$: agent's policy for $c \in C$, $R^{(c)}_{t:t+n} = \sum_{i=1}^{n} \gamma^i r^{(c)}_{t+i}$,
  θ: set of parameters of π and $\pi^{(c)}$s.
- Output of auxiliary tasks are not directly used. Instead, it implicitly affects representation learned, and thereby affects long-term performance.

Auxiliary Control Tasks

- Pixel changes : Train agents to maximally change the pixels in each cell of an $n \times n$ non-overlapping grid placed over the input image.



Auxiliary Control Tasks

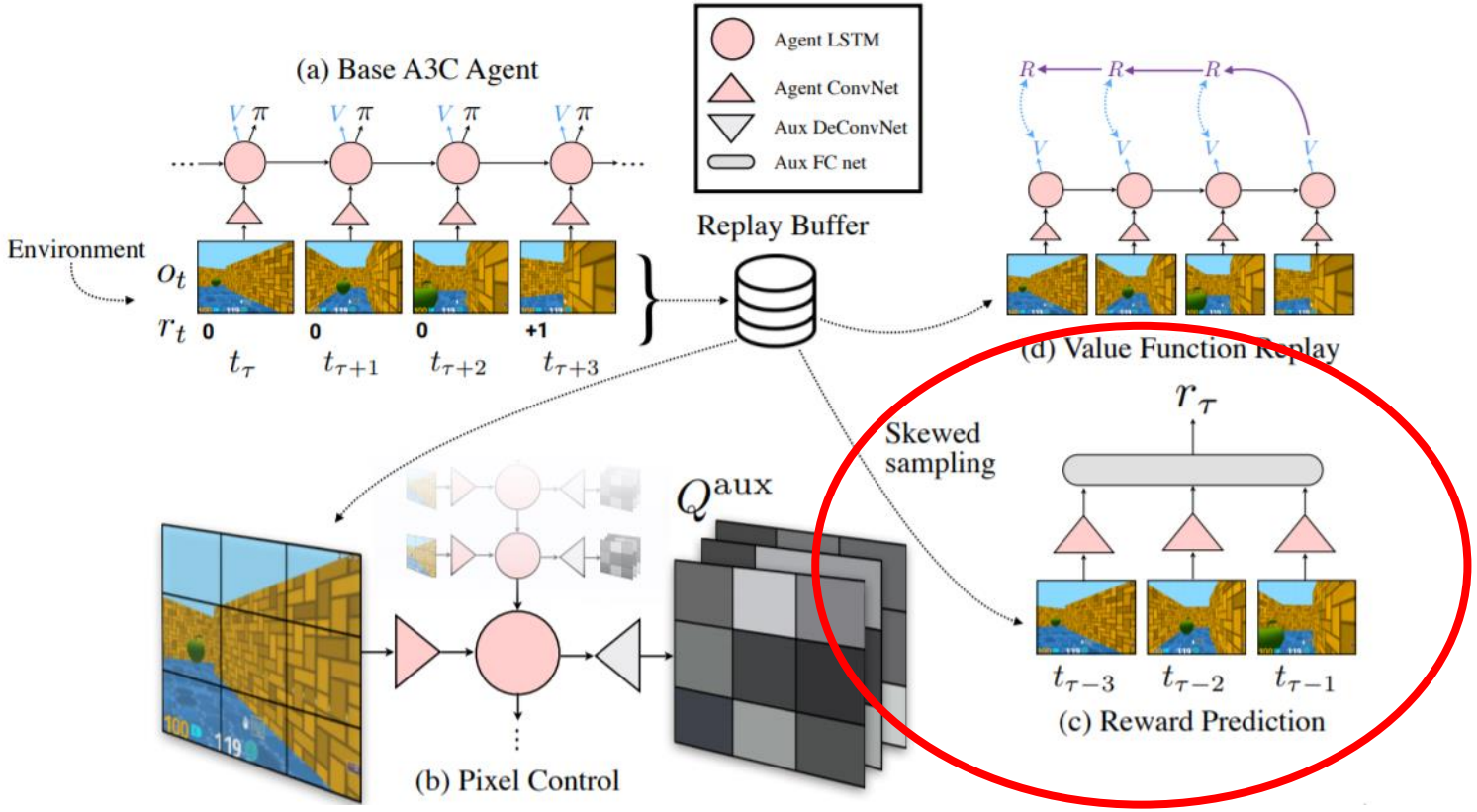
- Network features : The activation of any hidden unit of the agent's neural network can itself be an auxiliary reward. -> Train agents to maximally activate each of the units in a specific hidden layer.

Auxiliary Reward Tasks

- Agents must know how to maximize rewards
- Agents can do this through learning features that recognize states that lead to high reward and value.
- Agent with a good representation of reward states, will allow learning of good value functions, and in turn should allow the easy learning of a policy.

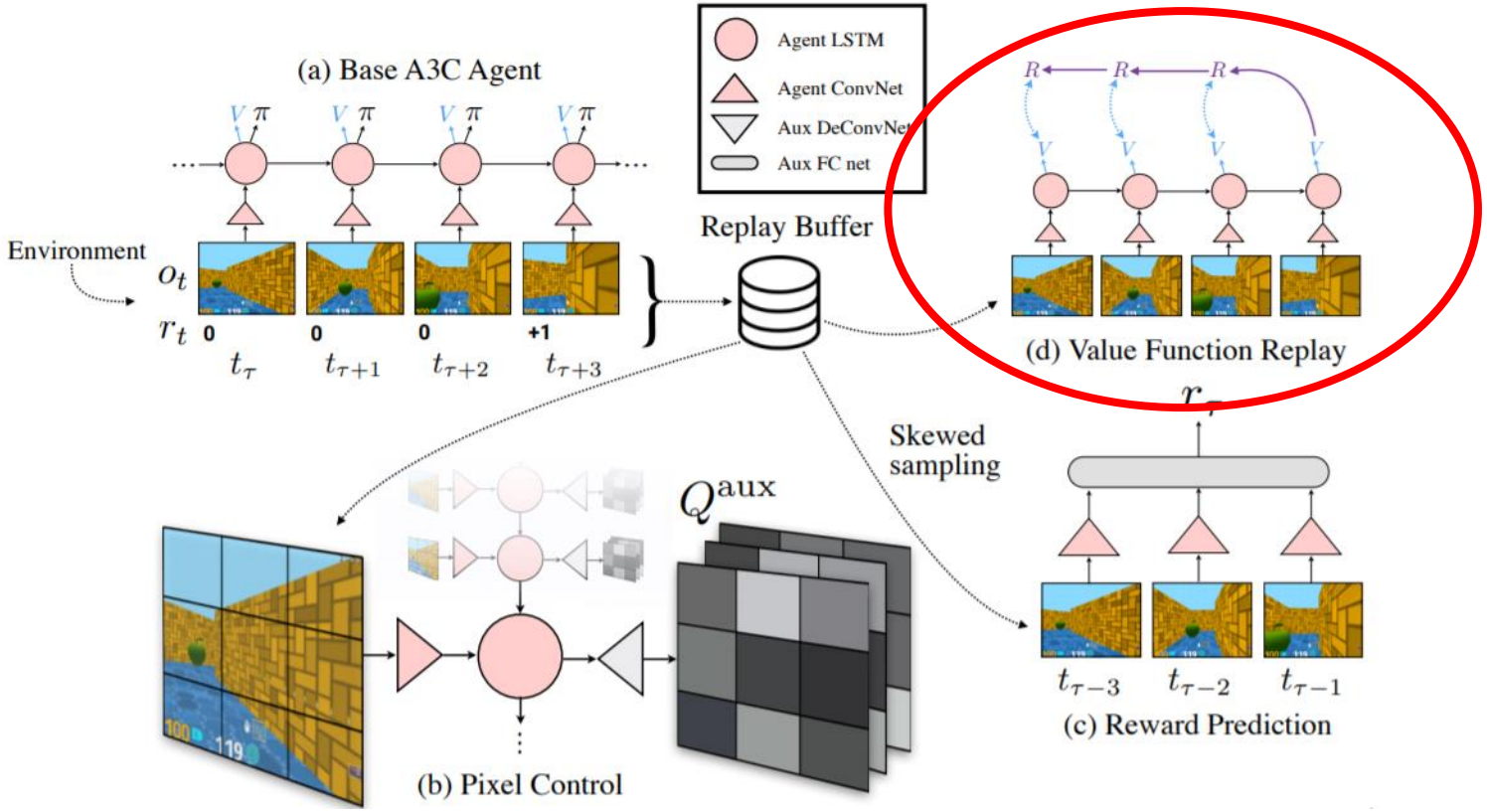
Auxiliary Reward Tasks

- Reward prediction : predict whether current state will give positive reward or not.
- Sample zero rewards and non-zero rewards equally, i.e. $P(r \neq 0) = 0.5$.
- Crossentropy loss is used for classification.



Auxiliary Reward Tasks

- Value function replay : predict whether current state will give positive reward or not.
- Regression of value function on resampled recent historical sequences.
- Repetition of value function regression will lead to faster convergence.



UNREAL Agent

- Unsupervised Reinforcement and Auxiliary Learning agent
- UNREAL algorithm combines A3C algorithms and auxiliary tasks.

$$\mathcal{L}_{UNREAL}(\theta) = \mathcal{L}_{\mathrm{A3C}} + \lambda_{\mathrm{VR}}\mathcal{L}_{\mathrm{VR}} + \lambda_{\mathrm{PC}} \sum_{c} \mathcal{L}_{Q}^{(c)} + \lambda_{\mathrm{RP}}\mathcal{L}_{\mathrm{RP}}$$

- $\mathcal{L}_{A3C}$: A3C loss, minimized on-policy
- $\mathcal{L}_{VR}$: replayed value loss, optimized from replayed data
- $\mathcal{L}_{Q}$: auxiliary control losses, optimized off-policy from replayed data, by n-step Q-learning
- $\mathcal{L}_{RP}$: replayed value loss, optimized from rebalanced replay data

Results

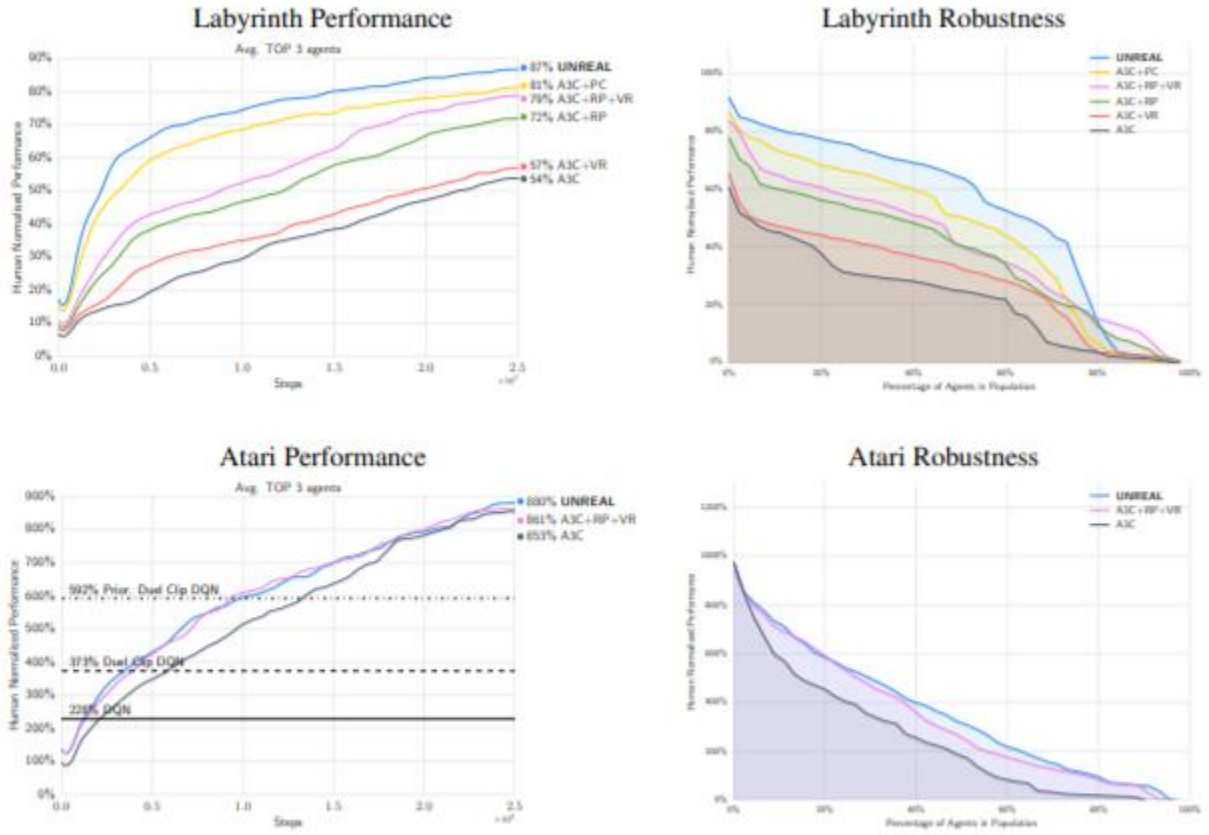


Figure 3: An overview of performance averaged across all levels on Labyrinth (Top) and Atari (Bottom). In the ablated versions RP is reward prediction, VR is value function replay, and PC is pixel control, with the *UNREAL* agent being the combination of all. *Left*: The mean human-normalised performance over last 100 episodes of the top-3 jobs at every point in training. We achieve an average of 87% human-normalised score, with every element of the agent improving upon the 54% human-normalised score of vanilla A3C. *Right*: The final human-normalised score of every job in our hyperparameter sweep, sorted by score. On both Labyrinth and Atari, the *UNREAL* agent increases the robustness to the hyperparameters (namely learning rate and entropy cost).

Results

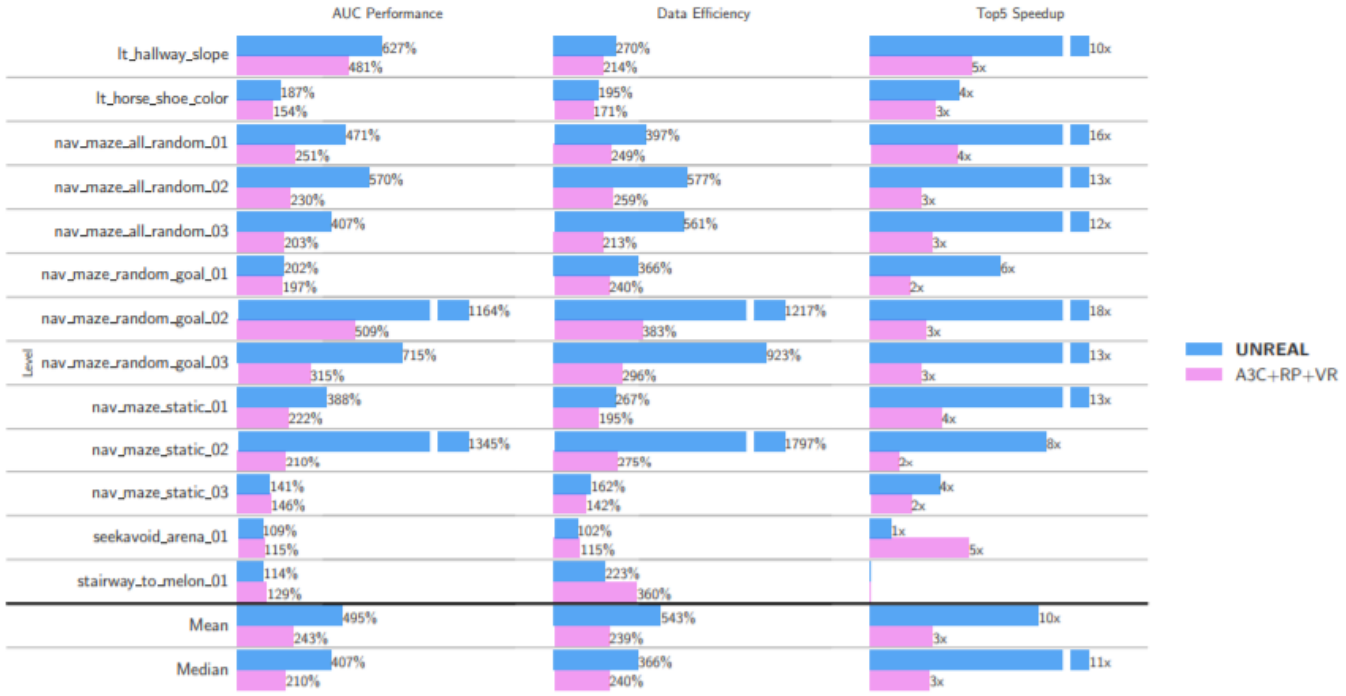


Figure 4: A breakdown of the improvement over A3C due to our auxiliary tasks for each level on Labyrinth. The values for A3C+RP+VR (reward prediction and value function replay) and *UNREAL* (reward prediction, value function replay and pixel control) are normalised by the A3C value. AUC Performance gives the robustness to hyperparameters (area under the robustness curve Figure 3 Right). Data Efficiency is area under the mean learning curve for the top-5 jobs, and Top5 Speedup is the speedup for the mean of the top-5 jobs to reach the maximum top-5 mean score set by A3C. Speedup is not defined for stairway_to_melon as A3C did not learn throughout training.

Results

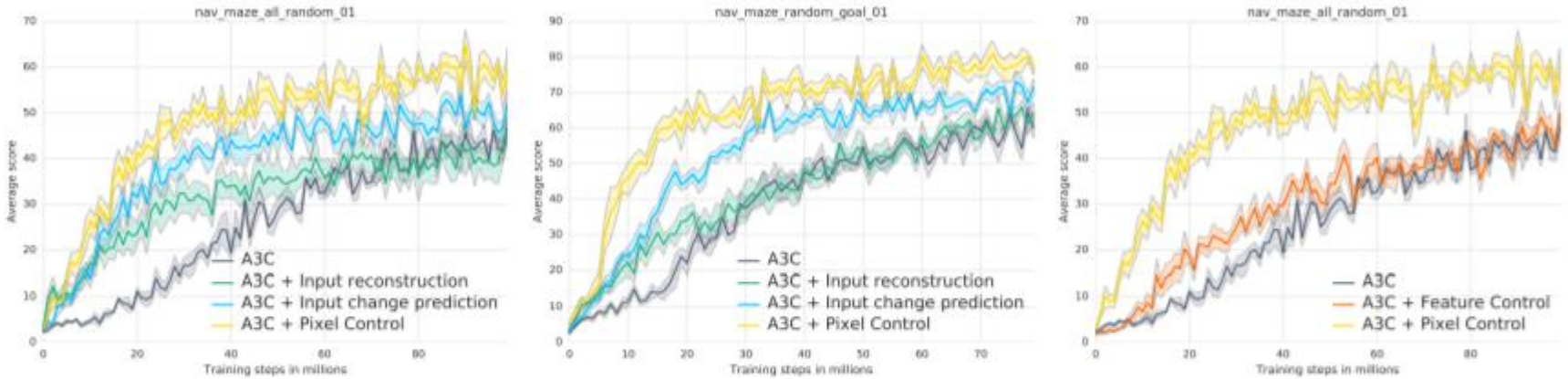


Figure 5: Comparison of various forms of self-supervised learning on random maze navigation. Adding an input reconstruction loss to the objective leads to faster learning compared to an A3C baseline. Predicting changes in the inputs works better than simple image reconstruction. Learning to control changes leads to the best results.

Conclusion

- Combining A3C algorithms and various auxiliary tasks drastically improves both data efficiency and robustness to hyperparameter settings.
- Achieved state-of-the-art in Labyrinth environment.
- Improved both the learning speed and the robustness in Atari environment.