Overview of Deep Reinforcement Learning

Pier Paolo Tarasco

27/01/2022

Survey Project Presentation

Outline

- DQN: How to play Atari Games
- GPS: Learning in the real world
- AlphaGo: Advanced MCTS to master the game of Go
- ICM: Designing the desire to explore

Design of the DQN agent

- Learns the Q function by the Bellman equation

$$Q_{i+1}(s,a) = E_{s'}\left[r + \gamma \max_{a'} Q_i(s',a') \mid s,a\right]$$

- Resolves instability problems by:
  - Experience Replay to break the correlation in the sequence of observations
  - Mantains an old copy of the Q(s,a) to use as guide in the SL and update it every C steps

$$L(\theta_i) = E_{s,a,r,s'}[(r + \gamma \ \max_{a'} Q(s',a',\theta_i^-) - Q(s,a;\theta_i))^2]$$

$$\nabla_{\theta_i} L(\theta_i) = E_{s,a,r,s'}[(r + \gamma \ \max_{a'} Q(s',a',\theta_i^-) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a,\theta_i)]$$

  - Normalize rewards [-1; 1]
  - Clip error term to [-1; 1]

Model Architecture

210x160

84x84x4

Deep CNN → Values for each action

- For each game agent experiences 50 million frames → 38 days of game experience!
- Exploration controlled by ϵ-greedy
- The action is selected once every 4 frames and with the last action repeated 4 times.

Results

- Not able to learn long-term behaviour, for example it cannot solve the Montazuma game
- Requires a lot of samples

Game	Linear	Without replay Without target	Without replay with target	With replay without target	DQN (With replay & with target)
Breakout	3	3.2	10.2	240.7	316.8
Enduro	62	29.1	141.9	831.4	1006.3
River Raid	2346.9	1453	2867.7	4102.8	7446.6
Seaquest	656.9	275.8	1003	822.6	2894.4
Space Invaders	301.3	302	373.2	826.3	1088.9

GPS: Guided Policy Search

- Learn a complex policy $\pi_\theta(u\,|o)$ represented by a NN by interacting with the real world
- $\pi_\theta(u\,|o)$ maps images to actions, how to avoid deploying a partially trained policy to a real robot?
- Train a simpler policy $p(u\,|x)$ on the full state then use SL to match $\pi_\theta(u\,|o)$ and $p(u\,|x)$

$$\min_{p,\pi_\theta} E_\pi[l(\tau)]$$

$$\text{subject to}$$

$$E_{p(u_t|x_t)p(x_t)}[u_t] = E_{\pi_\theta(u_t|x_t)p(x_t)}[u_t]$$

(B)ADMM optimization algorithm

Solved by simple SGD

$$\theta \leftarrow \arg\min_{\theta} \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)} \left[\mathbf{u}_t^{\mathrm{T}} \lambda_{\mu t}\right] + \nu_t \phi_t^{\theta}(\theta, p)$$

$$p \leftarrow \arg\min_{p} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)} \left[\ell\left(\mathbf{x}_t, \mathbf{u}_t\right) - \mathbf{u}_t^{\mathrm{T}} \lambda_{\mu t}\right] + \nu_t \phi_t^{p}(p, \theta)$$

$$\lambda_{\mu t} \leftarrow \lambda_{\mu t} + \alpha \nu_t \left(E_{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)} \left[\mathbf{u}_t\right] - E_{p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)} \left[\mathbf{u}_t\right]\right)$$

- The original problem is now rewritten into two minimizations of the Lagrangians followed by an update of the lagrangian multipliers
- Iterative process that converges to a local optimal solution

Trajectory optimization under hidden dynamics

$$p(u_t|x_t) = \mathcal{N}(K_t x_t + k_t, C_t)$$

$$p(x_{t+1}|x_t, u_t) = \mathcal{N}(f_{xt} x_t + f_{ut} u_t + f_{ct}, F_t)$$

- Learn the dynamics $p(x_{t+1}|x_t, u_t)$ using simple linear regression on the data generated by running the old policy on the robot
- Learn $p(u_t|x_t)$ using the LQR method
- Limit the change of $p(u_t|x_t)$ by bounding the KL-divergence

$$\min_{p(\tau) \in \mathcal{N}(\tau)} \mathcal{L}_p(p, \theta) \text{ s.t. } D_{\mathrm{KL}}(p(\tau) \| \hat{p}(\tau)) \leq \epsilon.$$

Spatial Softmax

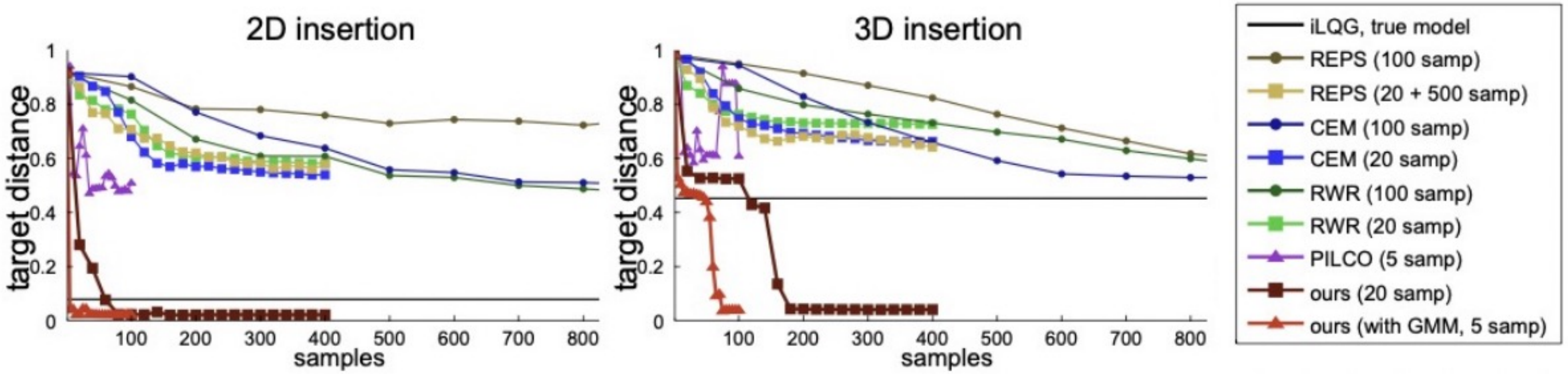
- The first layers of $\pi_\theta(u\,|o)$ are classic convolution layers
- We then use a novel operator Spatial Softmax to extract the position in the image of the learned features
- We concatenate the coordinates with the Robot State followed by a NN that predicts the torque forces for each of the 7 joints

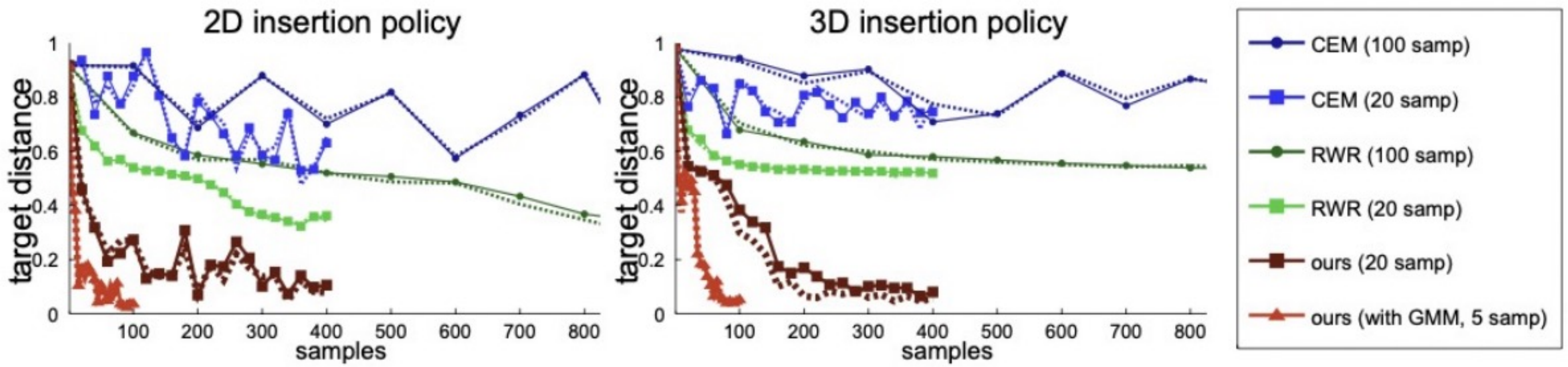
$$s_{cij} = \exp(a_{cij}) / \sum_{i',j'} \exp(a_{ci'j'})$$

$$f_{cx} = \sum_{ij} s_{cij} x_{ij} \qquad f_{cy} = \sum_{ij} s_{cij} y_{ij}$$

network architecture	test error (cm)
softmax + feature points (**ours**)	**1.30 ± 0.73**
softmax + fully connected layer	2.59 ± 1.19
fully connected layer	4.75 ± 2.29
max-pooling + fully connected	3.71 ± 1.73

Experiments





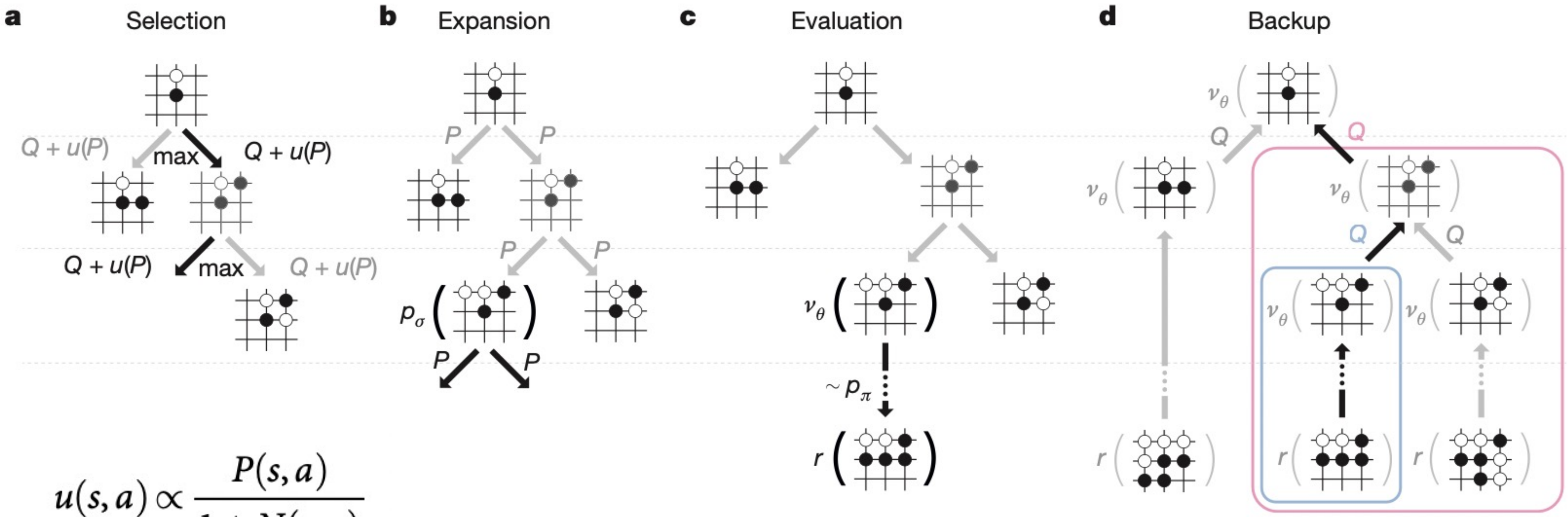
AlphaGo: Play Go at a professional level

- Zero-sum games can be solved optimally by Searching in the Game Tree
- Learn by copying humans
  - Trained on 30 million moves achieves 57% test accuracy
  - Fast rollout: reduced computational time but drops test accuracy to 24%
- Improve by self-play
  - Beats the previous policy 80% of the time $\Delta\rho \approx \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t$
- Learn the value function
  - Predicts the outcome of the RL policy
- Search for moves with MCTS

Architecture of the Policy and Value networks

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

MCTS



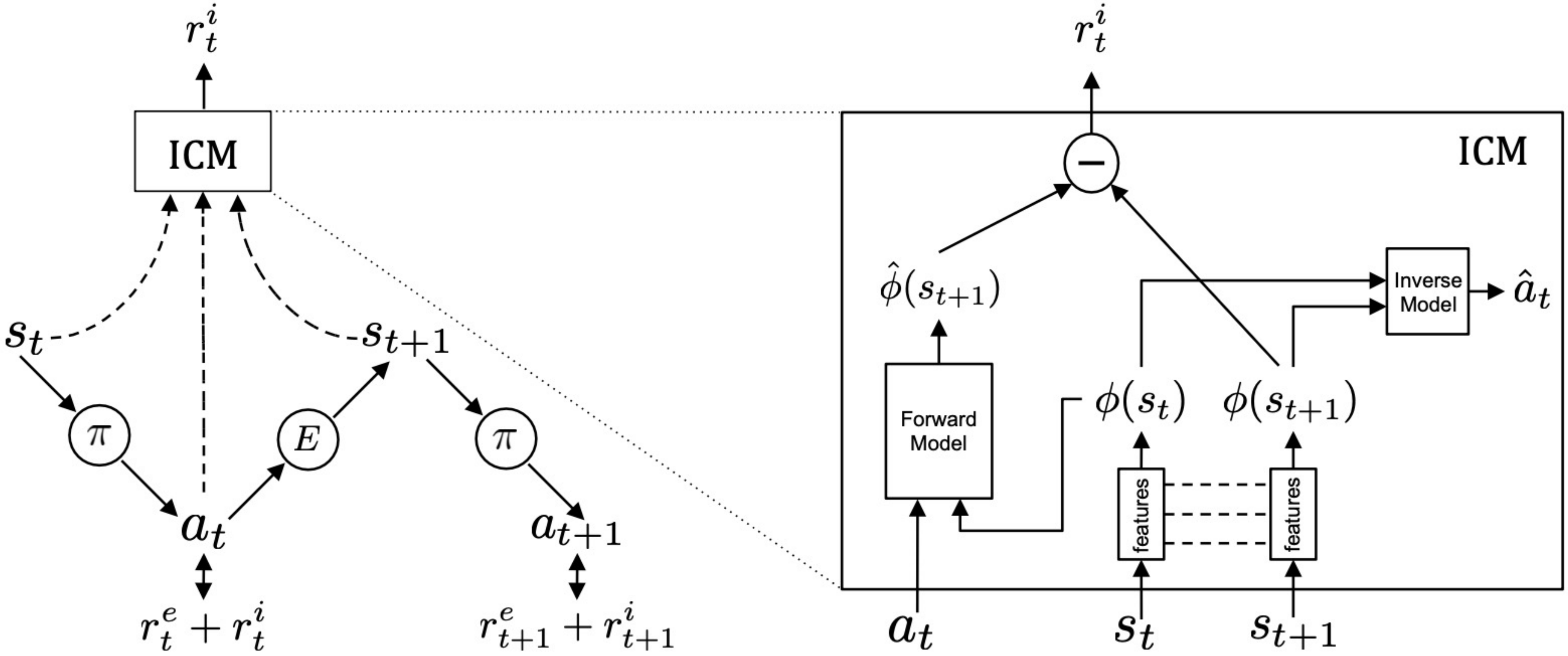
$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

Results

Short name	Computer Player	Version	Time settings	CPUs	GPUs	KGS Rank	Elo
α_{rvp}^{d}	Distributed AlphaGo	See Methods	5 seconds	1202	176	–	3140
α_{rvp}	AlphaGo	See Methods	5 seconds	48	8	–	2890
CS	CrazyStone	2015	5 seconds	32	–	6d	1929
ZN	Zen	5	5 seconds	8	–	6d	1888
PC	Pachi	10.99	400,000 sims	16	–	2d	1298
FG	Fuego	svn1989	100,000 sims	16	–	–	1148
GG	GnuGo	3.8	level 10	1	–	5k	431
CS_4	CrazyStone	4 handicap stones	5 seconds	32	–	–	2526
ZN_4	Zen	4 handicap stones	5 seconds	8	–	–	2413
PC_4	Pachi	4 handicap stones	400,000 sims	16	–	–	1756

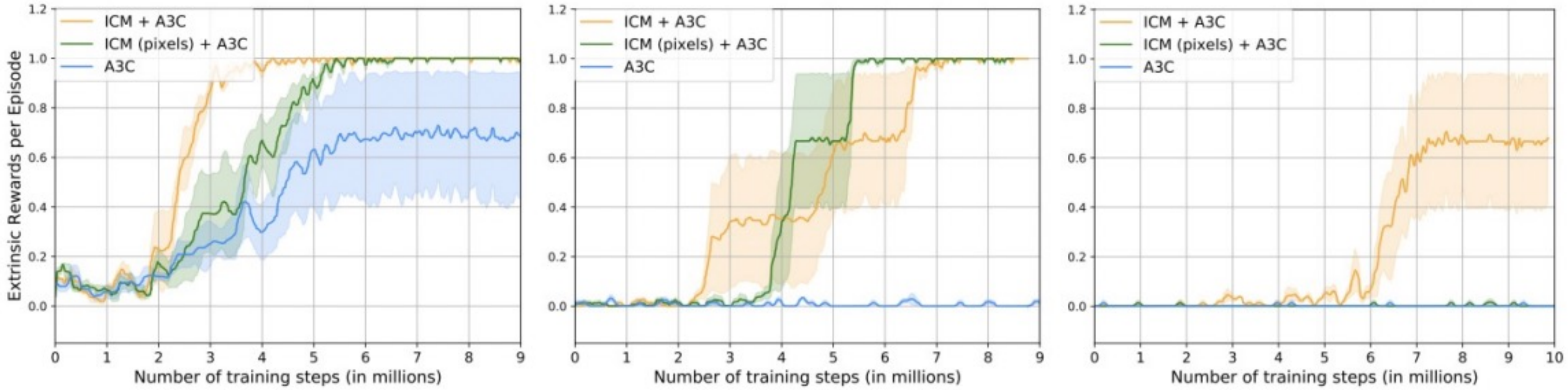
Short name	Policy network	Value network	Rollouts	Mixing constant	Policy GPUs	Value GPUs	Elo rating
α_{rvp}	p_σ	v_θ	p_π	$\lambda = 0.5$	2	6	2890
α_{vp}	p_σ	v_θ	–	$\lambda = 0$	2	6	2177
α_{rp}	p_σ	–	p_π	$\lambda = 1$	8	0	2416
α_{rv}	$[p_\tau]$	v_θ	p_π	$\lambda = 0.5$	0	8	2077
α_{v}	$[p_\tau]$	v_θ	–	$\lambda = 0$	0	8	1655
α_{r}	$[p_\tau]$	–	p_π	$\lambda = 1$	0	0	1457
α_{p}	p_σ	–	–	–	0	0	1517

ICM: Intrinsic Curiosity Module



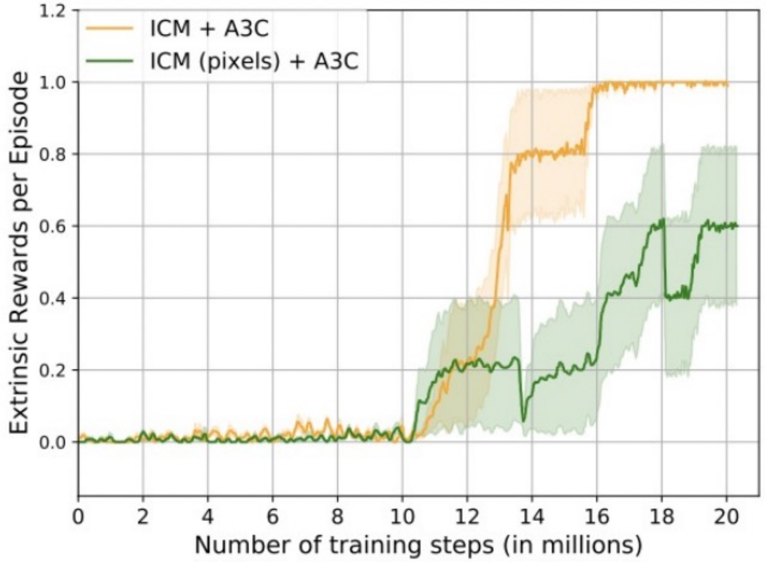
$$r_t^i = \frac{\mu}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$$

Sparsity

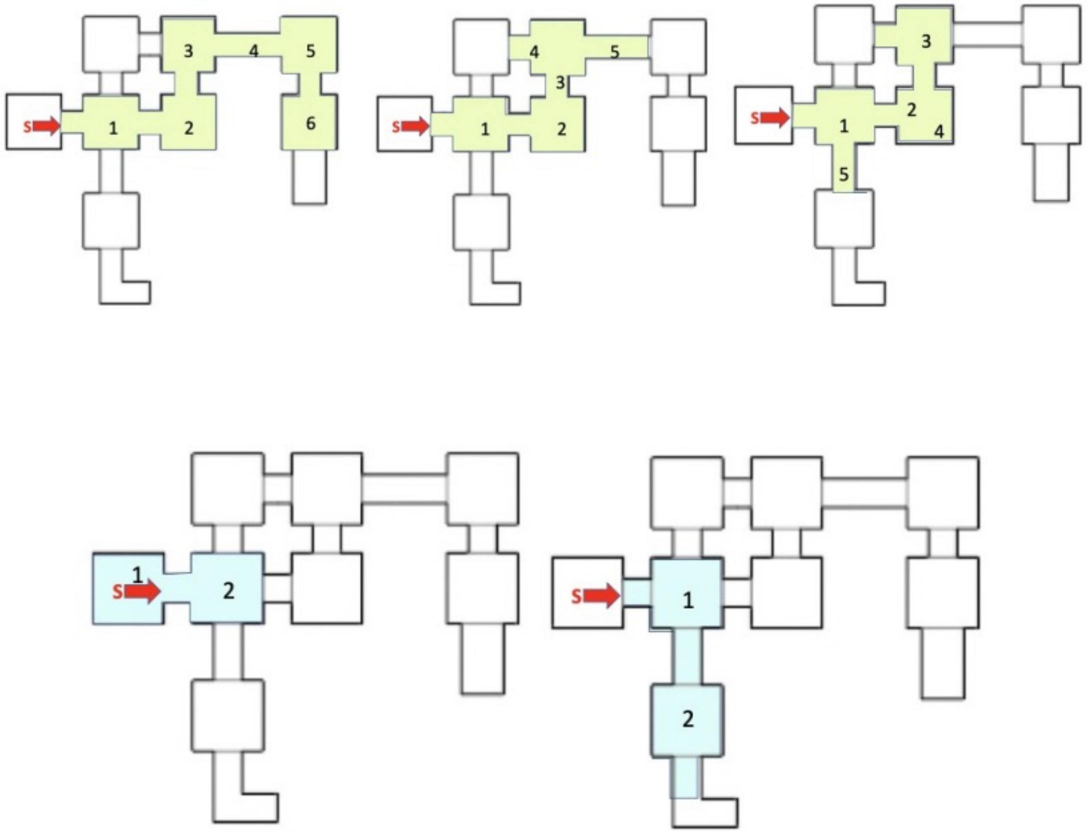


(a) "dense reward" setting (b) "sparse reward" setting (c) "very sparse reward" setting

Stochastic



Exploration Behaviour



References

- All the images and tables are taken by their corresponding papers:

[1] Sergey Levine et al. “End-to-End Training of Deep Visuomotor Policies”. In: *J. Mach. Learn. Res.* 17.1 (Jan. 2016), pp. 1334–1373. ISSN: 1532-4435.

[2] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836. URL: http://dx.doi.org/10.1038/nature14236.

[3] Deepak Pathak et al. “Curiosity-Driven Exploration by Self-Supervised Prediction”. In: ICML’17 (2017), pp. 2778–2787.

[4] David Silver et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

Thank you for your attention!