

STRAW

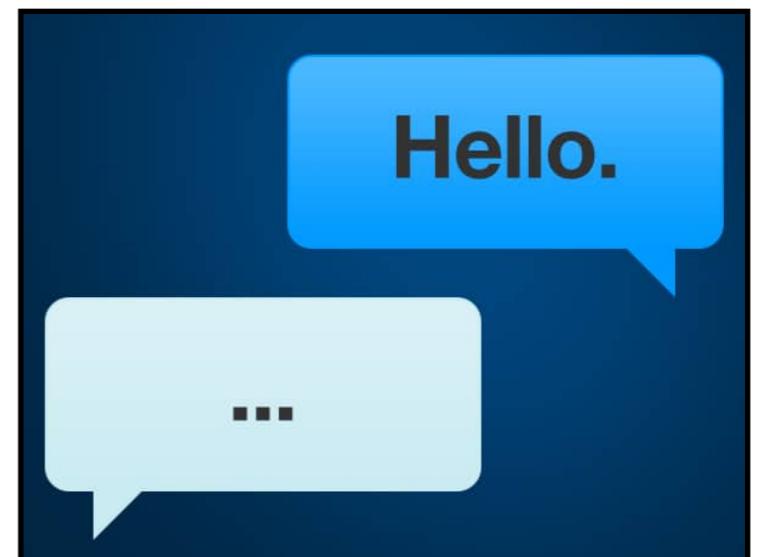
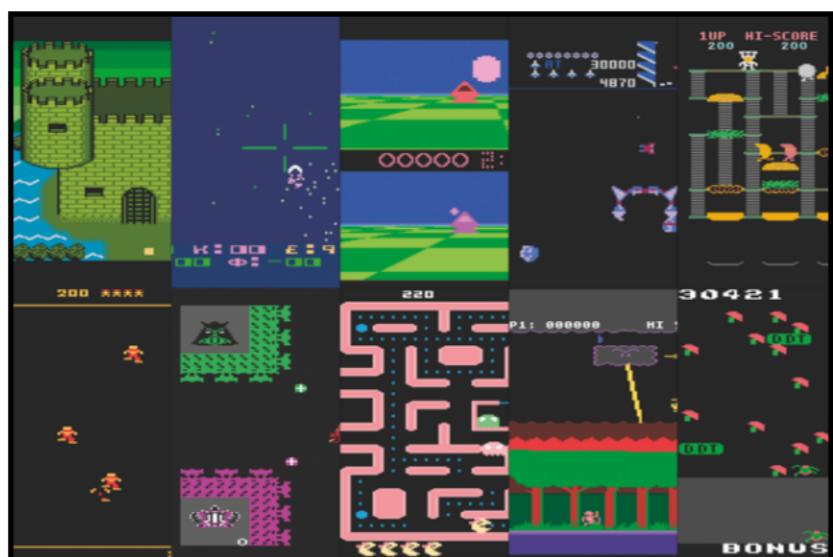
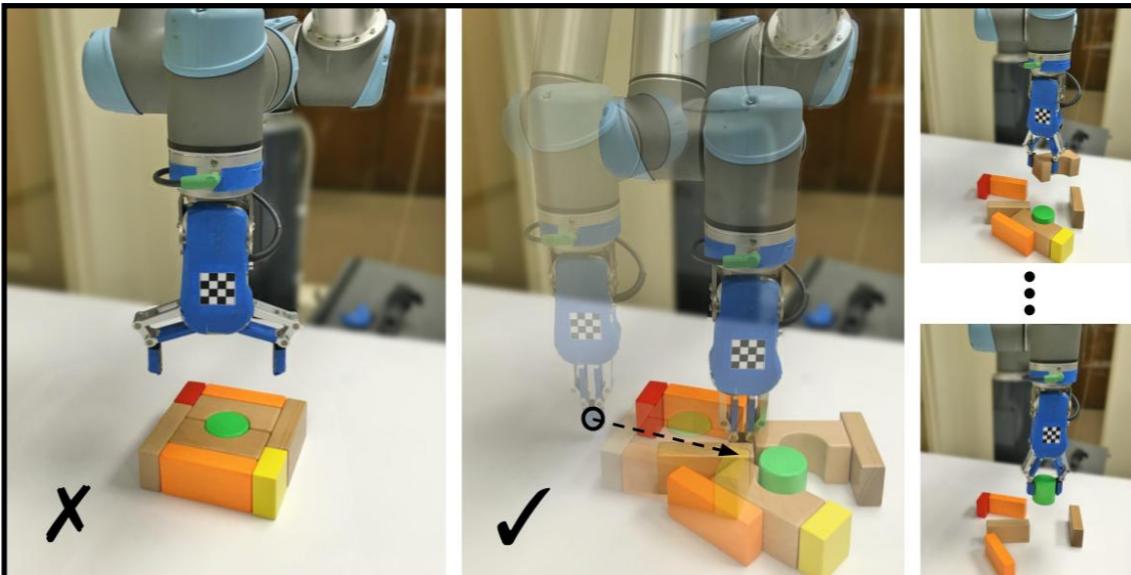
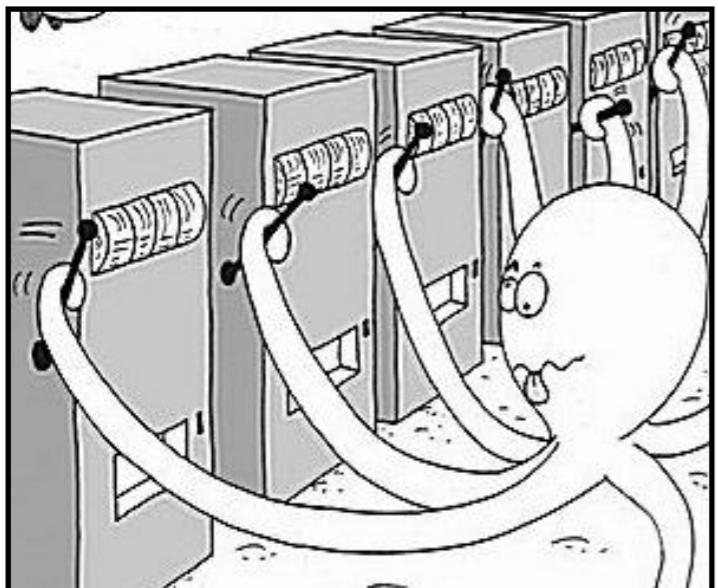
Strategic attentive writer for learning macro-actions

Soboleva Natalia, 151

Quick Reminder

What is reinforcement learning

Agent interacts with environment & We get feedback on agent performance



Underlying Math

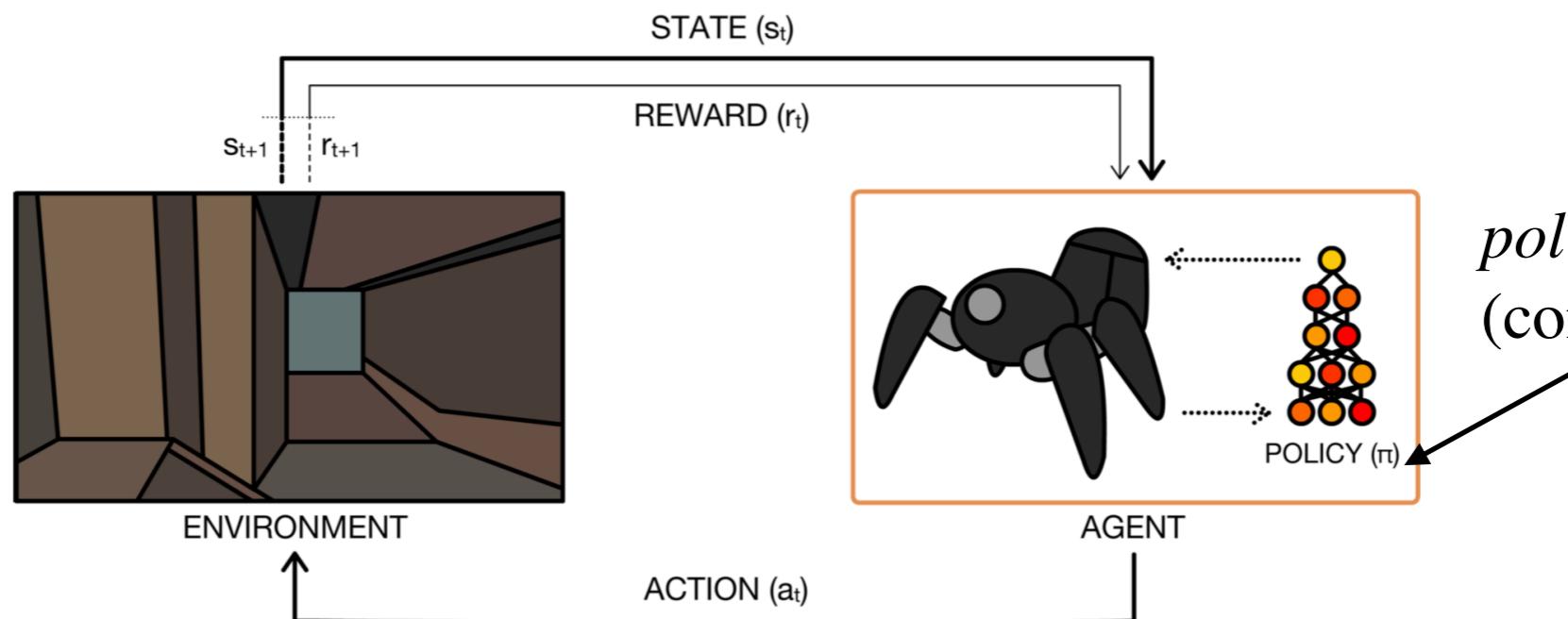
Markov Decision Processes (MDPs)

+ Observations

= *Partially Observed
Markov Decision
Processes (POMDPs)*

$$s_t \in S, a_t \in A, P(s_{t+1} | s_t, a_t), R : S \times A \rightarrow \mathbb{R}$$

$$s_t \in S, a_t \in A, P(s_{t+1} | s_t, a_t), R : S \times A \rightarrow \mathbb{R}$$
$$o_t \in O, P(o_t | s_{t+1}, a_t)$$



Source: *A Brief Survey of Deep Reinforcement Learning* (<https://arxiv.org/pdf/1708.05866.pdf>)

RL Evolution

Value Functions

Dynamic Programming

Q-learning / SARSA

Sampling

Monte Carlo policy evaluation

TD learning

Policy Search

Policy Gradients

DQN

Actor-critic Methods

Multi-agent RL

Model-based RL

Hierarchical RL

Memory and Attention

Value Iterations

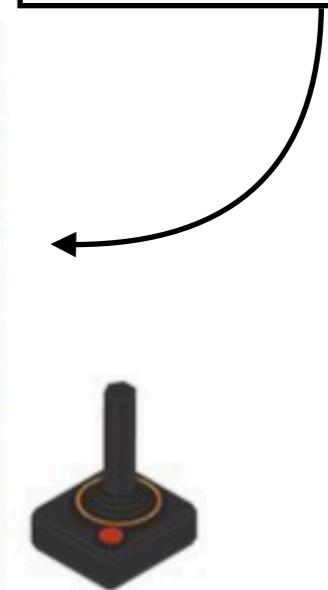
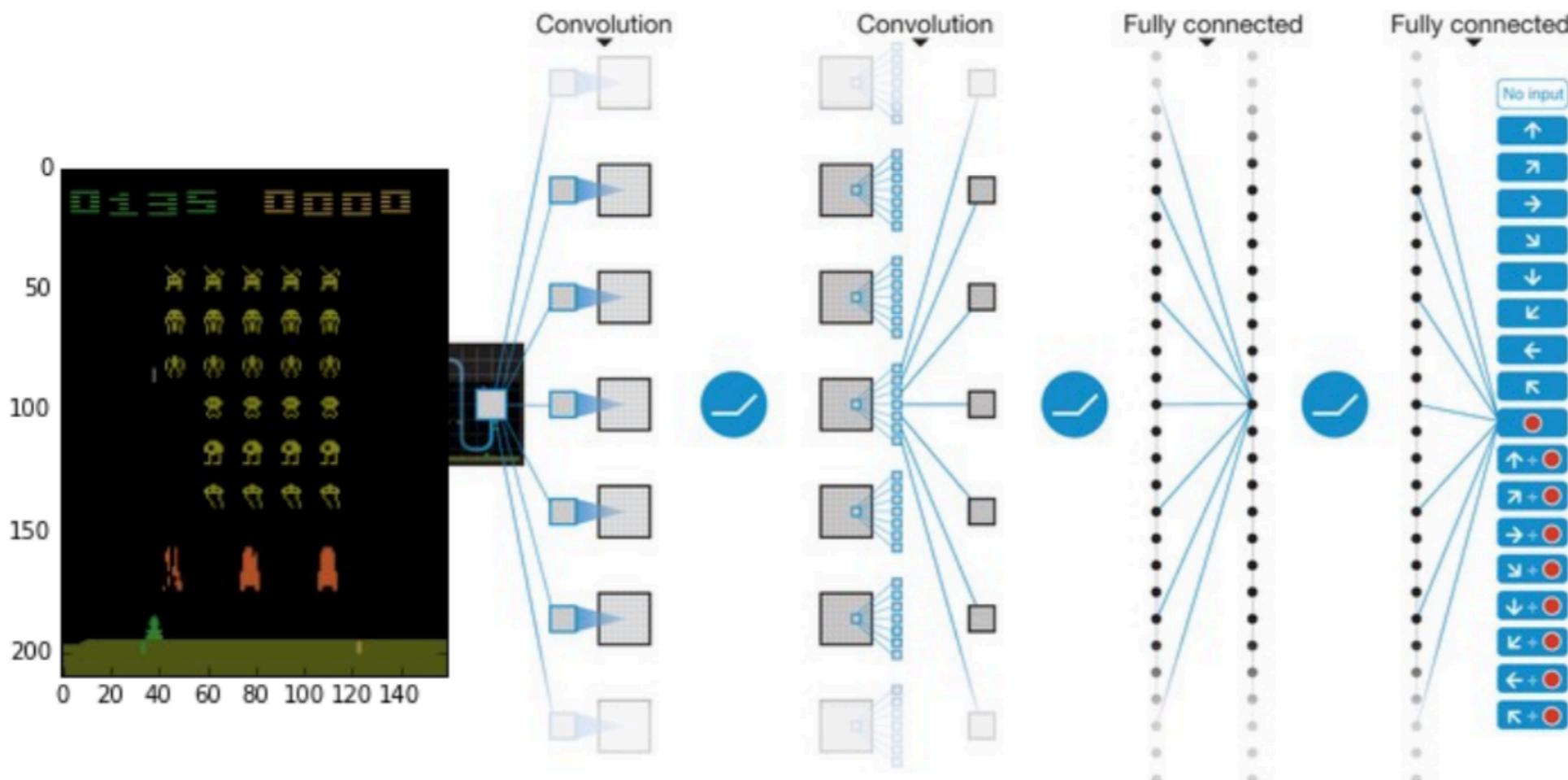
$$V(s) = \max_a [r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s, a)} V(s')]$$

$$Q(s, a) = \mathbb{E}'_s [r(s, a) + \gamma V(s')]$$



Going deeper

\mathcal{E} – greedy rule
(tune \mathcal{E} or use
probabilistic rule)



Policy gradient

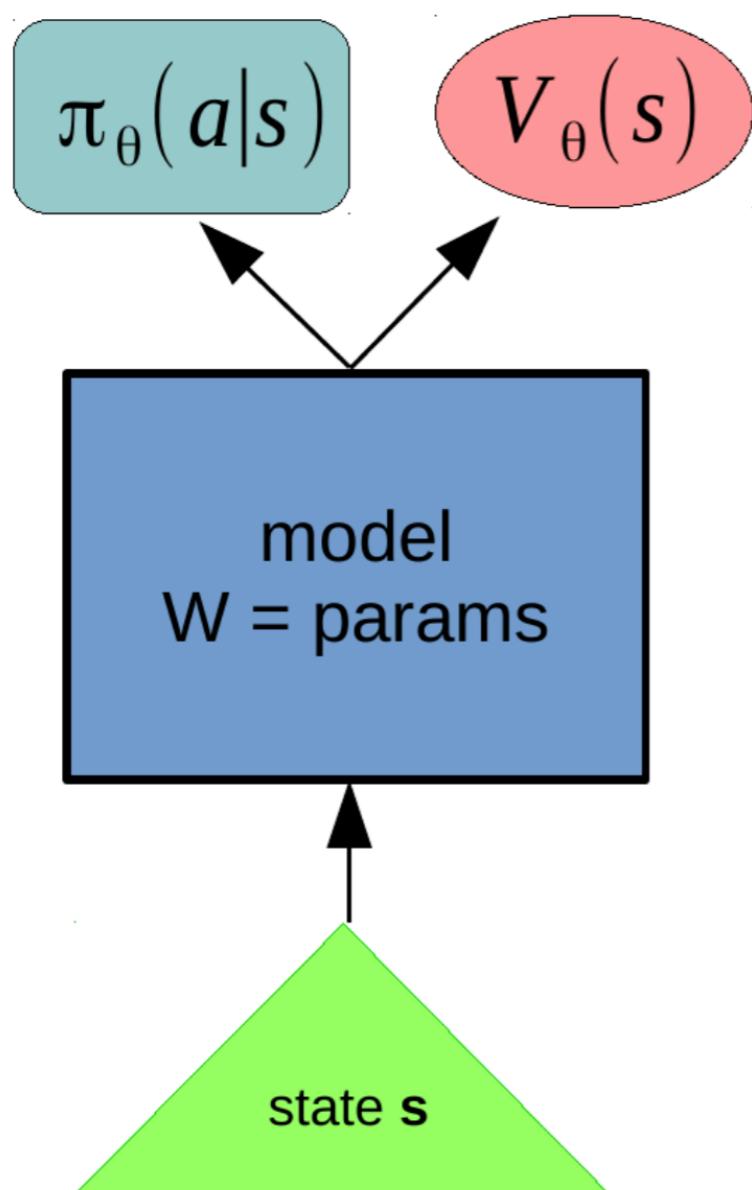
Learn

$$\pi_{\theta}(a | s) \rightarrow a$$

$$J = \mathbb{E}_{s,a} Q(s, a) = \int_s p(s) \int_a \pi_{\theta}(a | s) Q(s, a) da ds$$

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{a,s} \nabla \log \pi_{\theta}(a | s) Q(s, a)$$

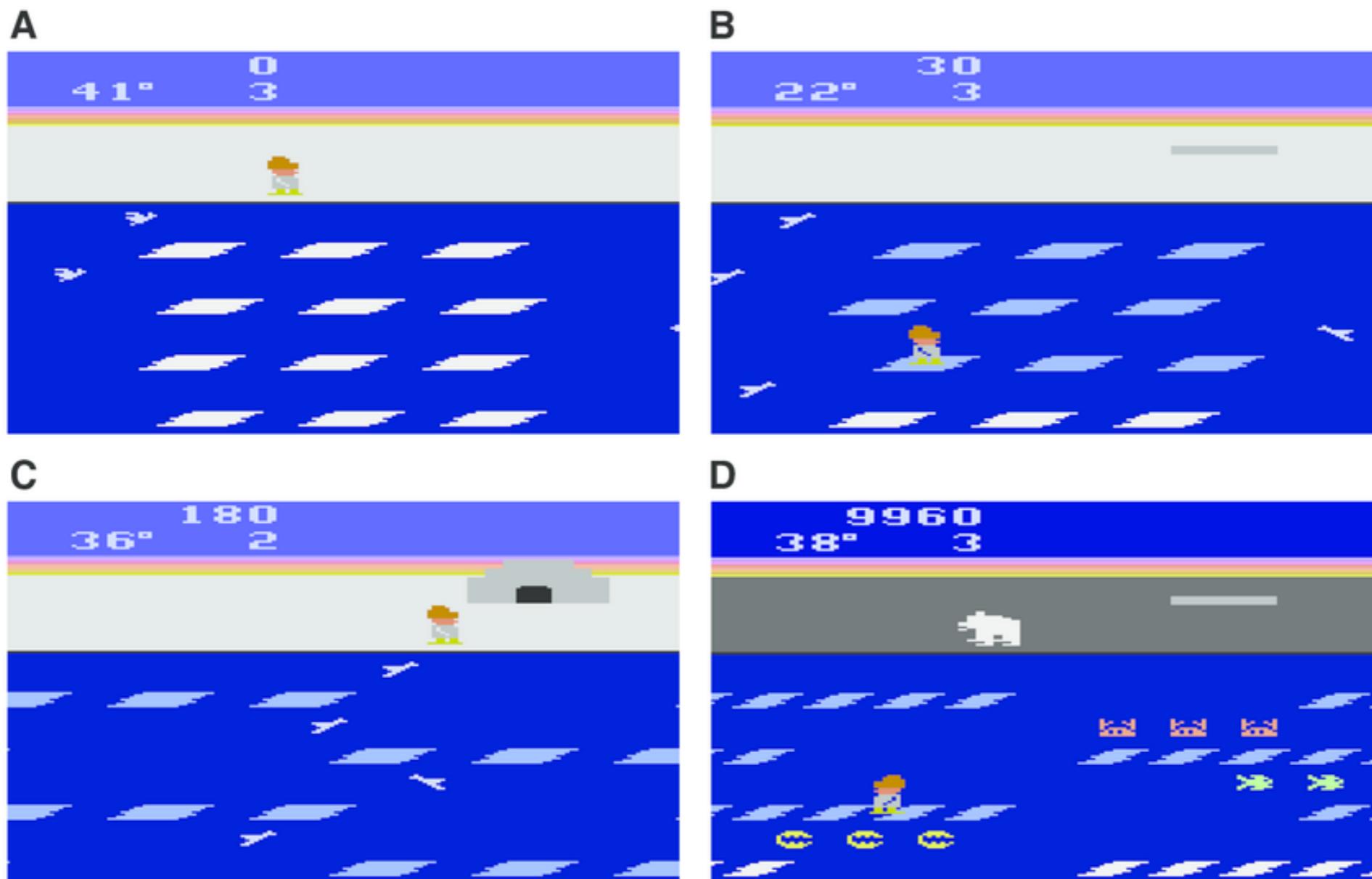
Advantage actor-critic



Learn both $\pi_\theta(a | s)$ and $V_\theta(s)$

Use $V_\theta(s)$ to learn $\pi_\theta(a | s)$ faster!

Frostbite Challenge



Source: <https://www.researchgate.net>

- A) The start of a level in Frostbite. The agent must construct an igloo by hopping between ice flows and avoiding obstacles such as birds. The floes are in constant motion (either left or right), making multi-step planning essential to success.
- B) The agent receives pieces of the igloo (top right) by jumping on the active ice floes (white), which then deactivates them (blue).
- C) At the end of a level, the agent must safely reach the completed igloo.
- D) Later levels include additional rewards (fish) and deadly obstacles (crabs, clams, and bears).

Problems with *options*

An *option* is a sub-policy with a termination condition, which takes in environment observations and outputs actions until the termination condition is met.

Needed:

- A. Policy-over-options
- B. Subgoals  **unsolved problem**
- C. “Pseudo-rewards”

STRategic Attentive Writer

Main Ideas:

- 1) STRAW learns ‘macro-actions’ – common action sequences – based on rewards.
- 2) STRAW also learns when to stick with the plan it has, and when it’s time to make a new plan.

action module

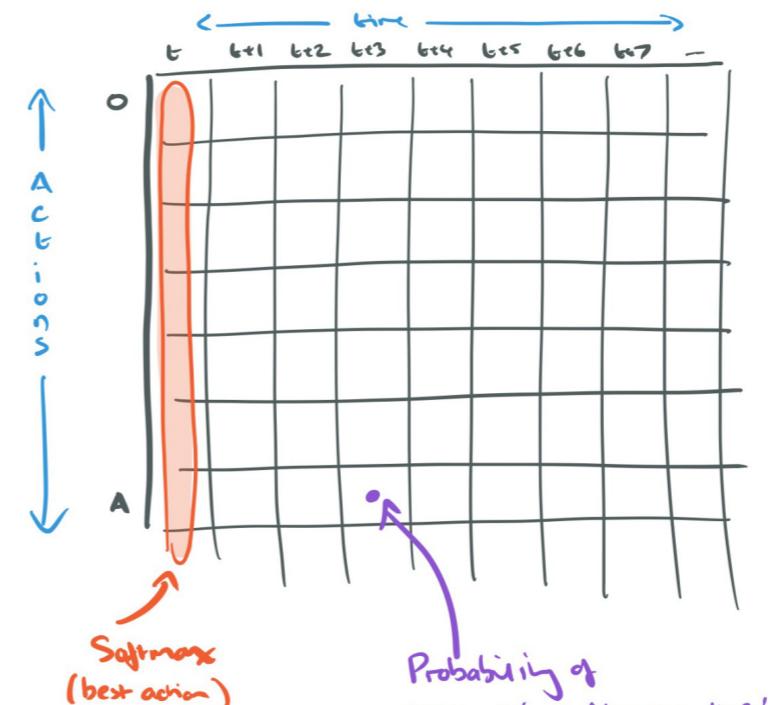
$$A^t \in \mathbb{R}^{A \times T}$$

A – the total number of possible actions

T – the maximum time horizon of a plan

$$A_{a,\tau}^t \propto P(a | \tau)$$

commitment module



STRategic Attentive Writer

Main Ideas:

- 1) STRAW learns ‘*macro-actions*‘ – common action sequences – based on rewards.
- 2) STRAW also learns when to stick with the plan it has, and when it’s time to make a new plan.

action module

$$A^t \in \mathbb{R}^{A \times T}$$

A – the total number of possible actions

T – the maximum time horizon of a plan

$$A_{a,\tau}^t \propto P(a | \tau)$$

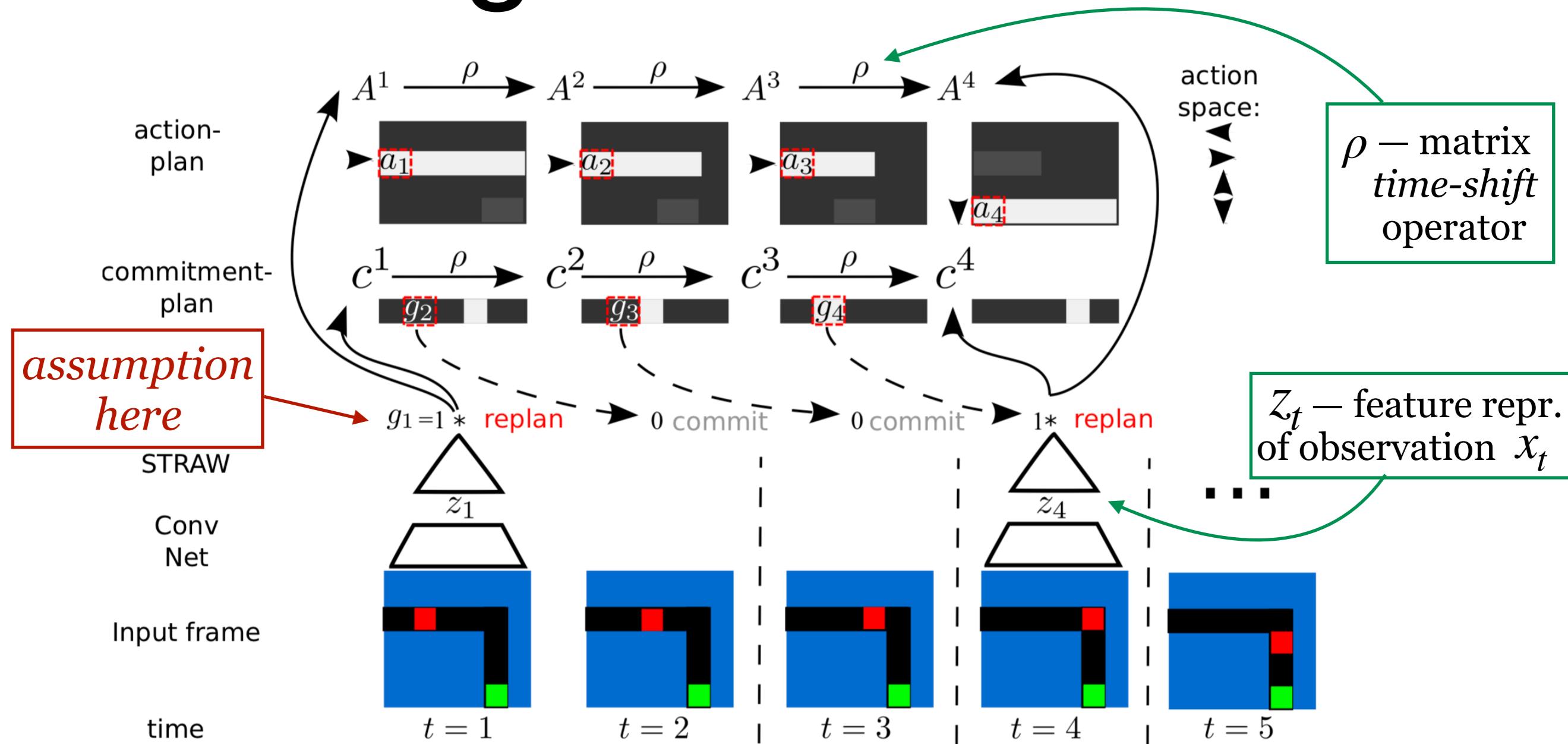
commitment module

$$c^t \in \mathbb{R}^{1 \times T}$$

Let g_t be a binary random variable distributed as follows: $g_t \sim c_1^{t-1}$

$$g_t = \begin{cases} 0, & \text{we are committed to the plans;} \\ 1, & \text{at this step the plans will be updated.} \end{cases}$$

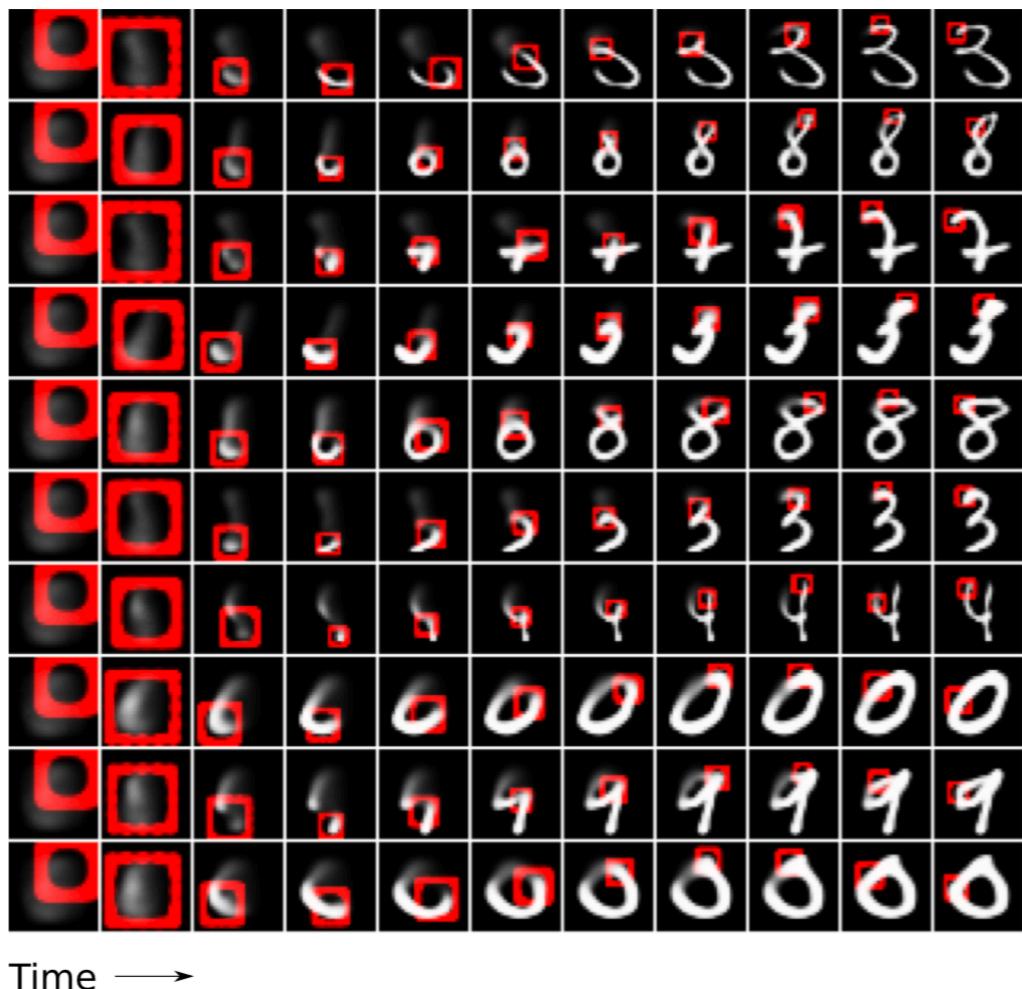
STRategic Attentive Writer



A frame is first passed through a convolutional network, acting as a feature extractor and then into STRAW. The top two rows depict the plans A and c . Given a gate g_t , STRAW either updates the plans (steps 1 and 4) or commits to them.

Attentive planning

Need to focus!



• • • ►

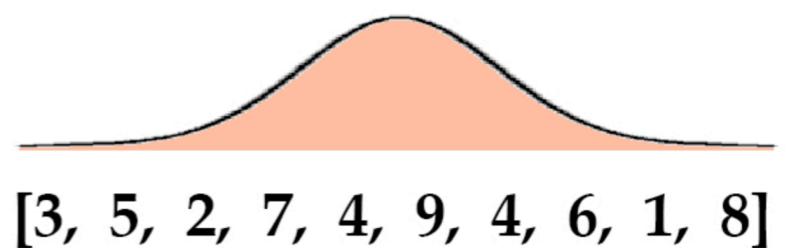
Mean average: all values hold the same influence

influence



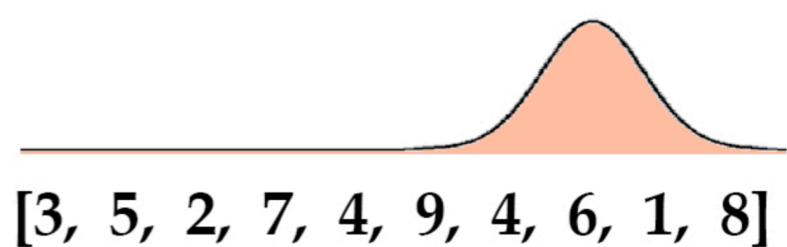
Gaussian filter: center values have more influence

influence



Gaussian filter can have any center / spread

influence



Source: DRAW: A Recurrent Neural Network For Image Generation.
(<https://arxiv.org/pdf/1502.04623.pdf>)

Source: <http://kvfrans.com/what-is-draw-deep-recurrent-attentive-writer>

Attentive planning

ψ – vector of attention parameters, i.e.: grid position, stride, and standard deviation of Gaussian filters

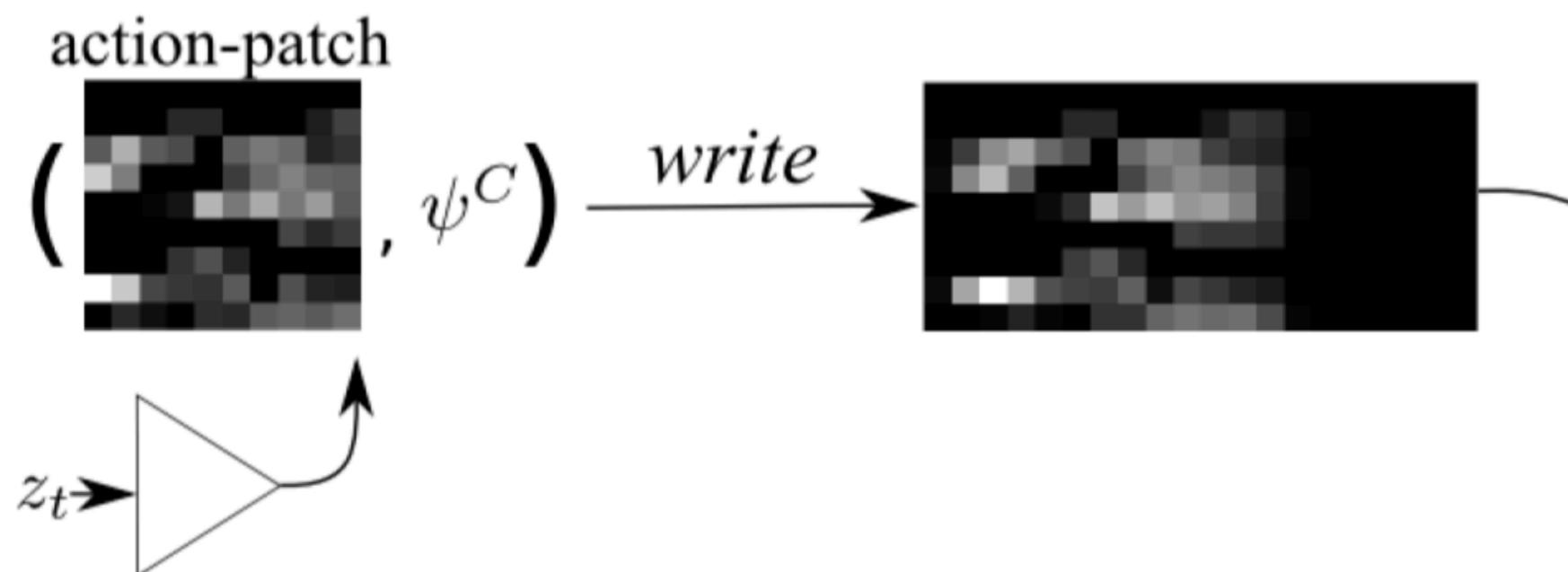
We define the attention operations as follows:

$$D = \text{write}(p, \psi_t^A), \quad \beta_t = \text{read}(A^t, \psi_t^A)$$

$$p \in \mathbb{R}^{A \times K}$$

$$\beta \in \mathbb{R}^{A \times K}$$

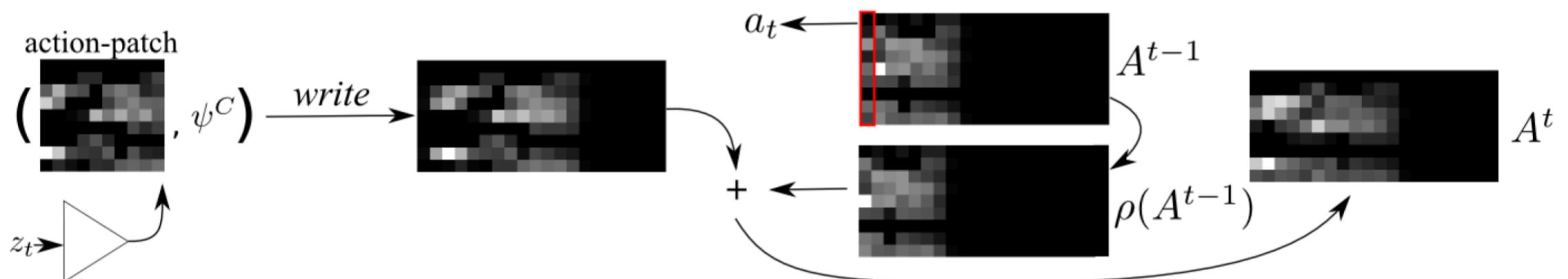
K – temporal resolution
of the patch



Action-plan update

Algorithm 1 Action-plan update

Input: $z_t, g_t, \mathbf{A}^{t-1}$
 Output: \mathbf{A}^t, a_t
if $g_t = 1$ **then**
 Compute attention parameters $\psi_t^A = f^\psi(z_t)$
 Attentively read the current state of the action-plan $\beta_t = \text{read}(\mathbf{A}^{t-1}, \psi_t^A)$;
 Compute intermediate representation $\xi_t = h([\beta_t, z_t])$
 Update $\mathbf{A}^t = \rho(\mathbf{A}^{t-1}) + g_t \cdot \text{write}(f^A(\xi_t), \psi_t^A)$
else // $g_t = 0$
 Update $\mathbf{A}^t = \rho(\mathbf{A}^{t-1})$ //just advance time
end if
 Sample an action $a_t \sim \text{SoftMax}(\mathbf{A}_0^t)$



Given z_t , the network produces an action-patch and attention parameters ψ_t^A . The write operation creates an update to the action-plan by scaling and shifting the action-patch according to ψ_t^A . The update is then added to $\rho(A^t)$.

Commitment-plan update

$g_t \sim \mathbf{c}_1^{t-1}$ if $g_t = 0$ then $\mathbf{c}^t = \rho(\mathbf{c}^{t-1})$ else $\psi_t^c = f^c([\psi_t^A, \xi_t])$

$\mathbf{c}_t = \text{Sigmoid}(\mathbf{b} + \text{write}(e, \psi^c));$

e – scalar set to random high value
(here: 40)

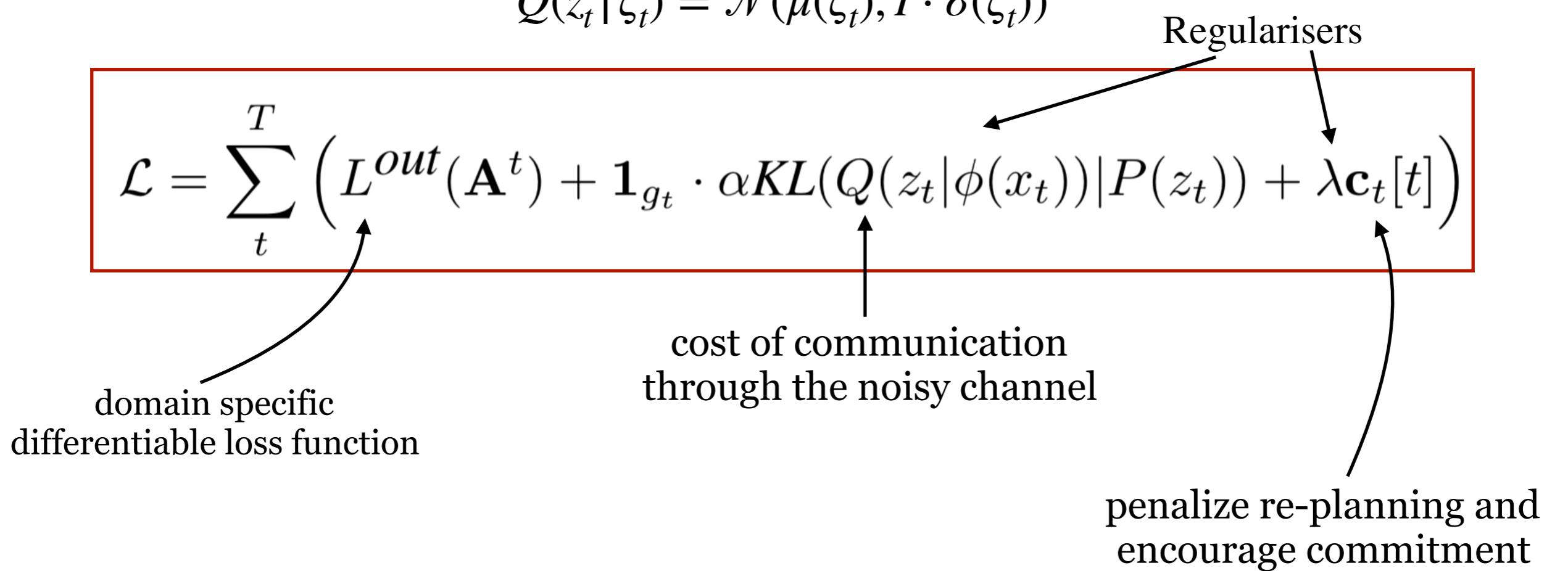
b – learnable bias, which defines the probability of
re-planning earlier than the step implied by ψ^c

STRAW-explorer & Learning

Adding noisy communication channel between the feature extractor and the STRAW planning modules. This noise generates randomness encouraging exploration of plan steps.

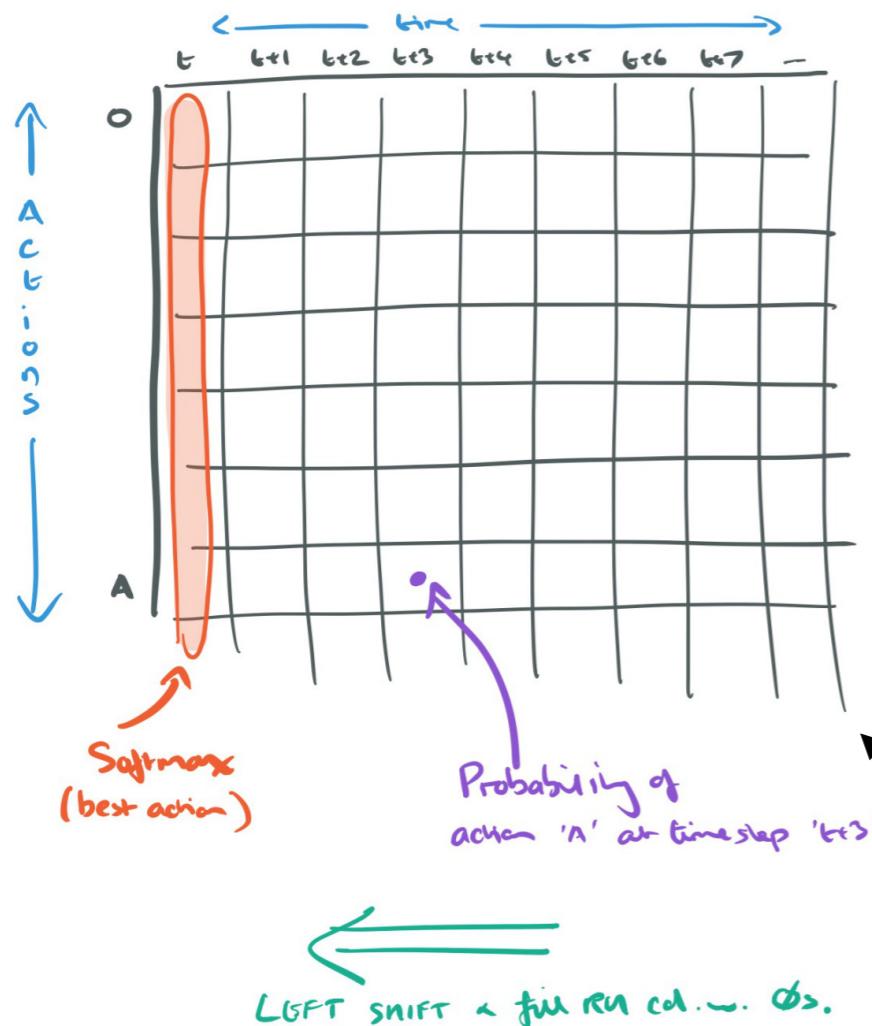
Let ζ_t be the activations of the last layer of the feature extractor.

$$Q(z_t | \zeta_t) = \mathcal{N}(\mu(\zeta_t), I \cdot \sigma(\zeta_t))$$



Asynchronous Advantage Actor-Critic (A3C)

$$\pi(a_t | x_t, \theta) = \text{SoftMax}(A_{\bullet 0}^t)$$



$$\nabla L^{out} = \nabla_{\theta} \log \pi(a_t | x_t; \theta) (R_t - V(x_t; \theta)) + \beta \nabla_{\theta} H(\pi(a_t | x_t; \theta))$$

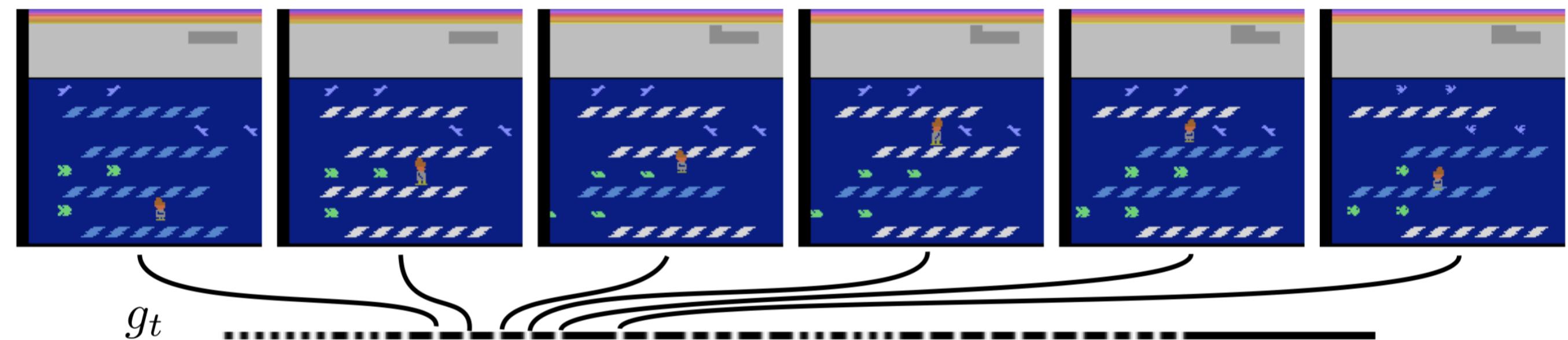
discounted reward sum

entropy of the policy,
which stimulates the
exploration of primitive actions

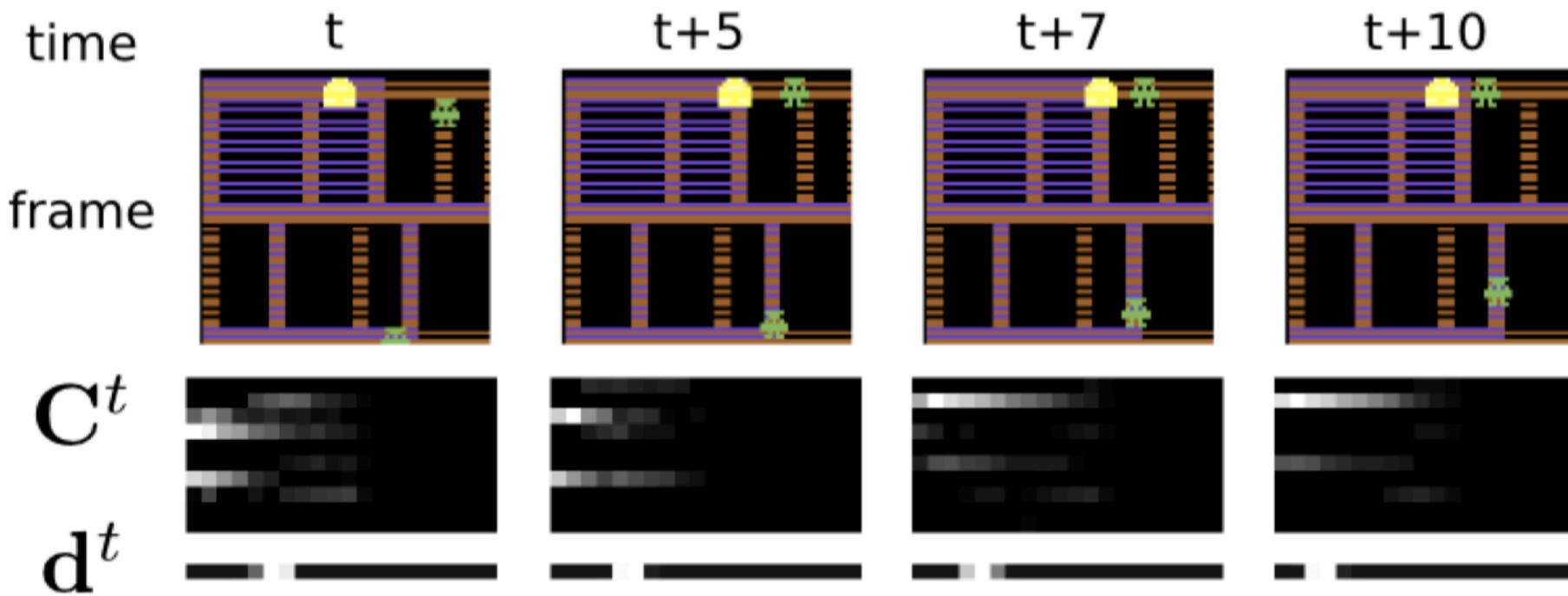
**add an auxiliary row
with value function
estimation $V_{\theta}(s)$**

What did we gain?

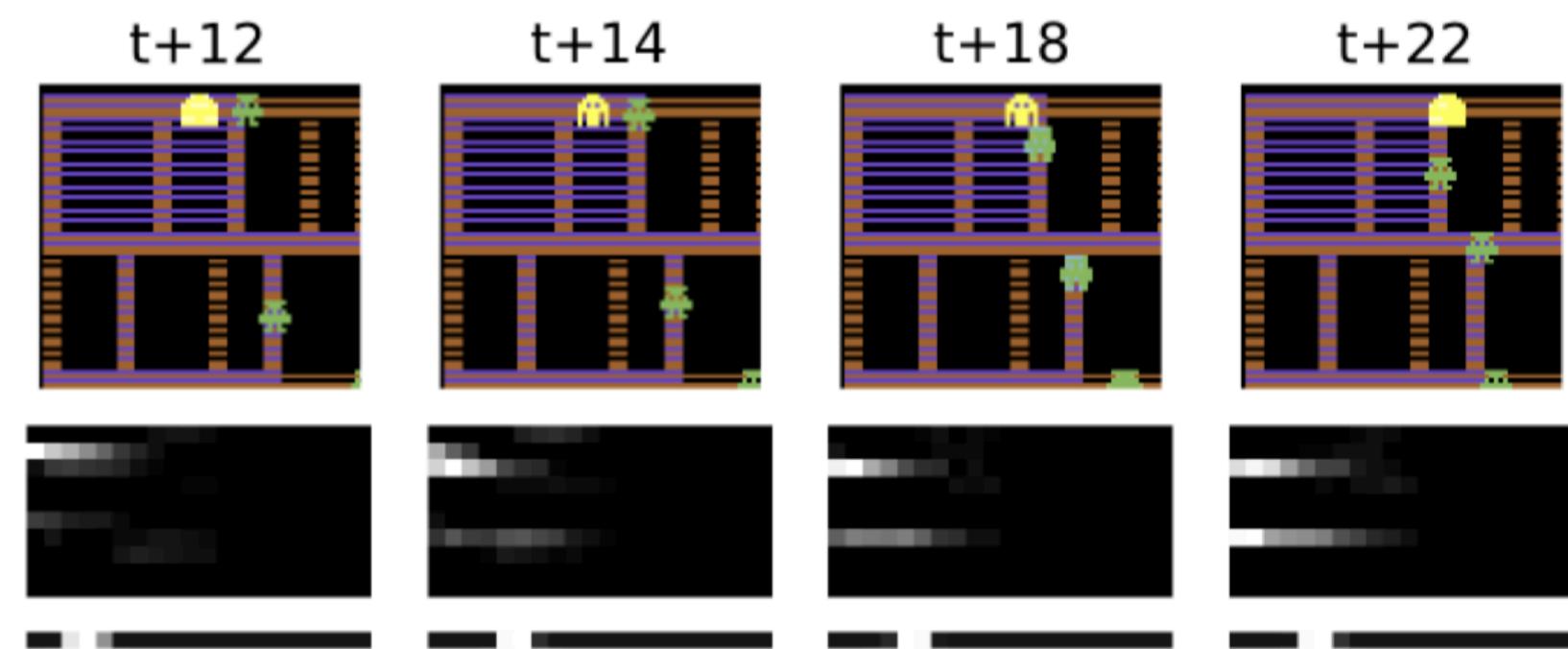
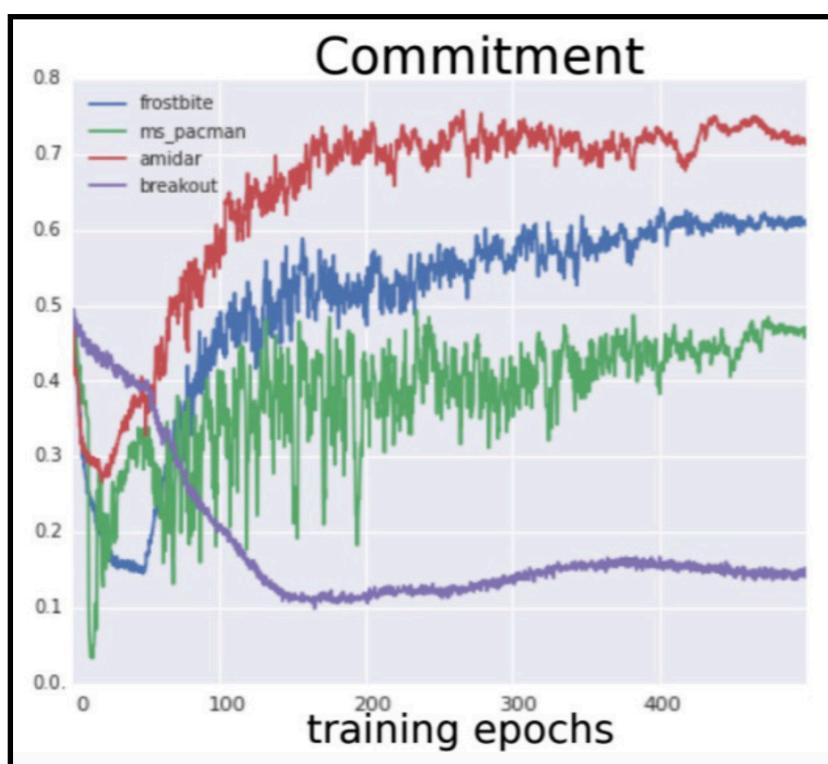
	Frostbite	Ms. Pacman	Q-bert	Hero	Crazy cl.	Alien	Amidar	Breakout
STRAWe	8108	6902	23892	36931	153327	3230	2022	386
STRAW	4394	6730	20933	36734	143803	2626	2223	344
LSTM	1281	4227	23121	36072	128932	2984	1392	644
FF	946	2014	22947	39929	115944	2300	1292	106



From top to bottom, the figure shows frames at which STRAW re-plans;
 g_t – white for 1, black for zero. Notice how macro-actions correspond to meaningful high-level actions like jumping from floe to floe and picking fish.

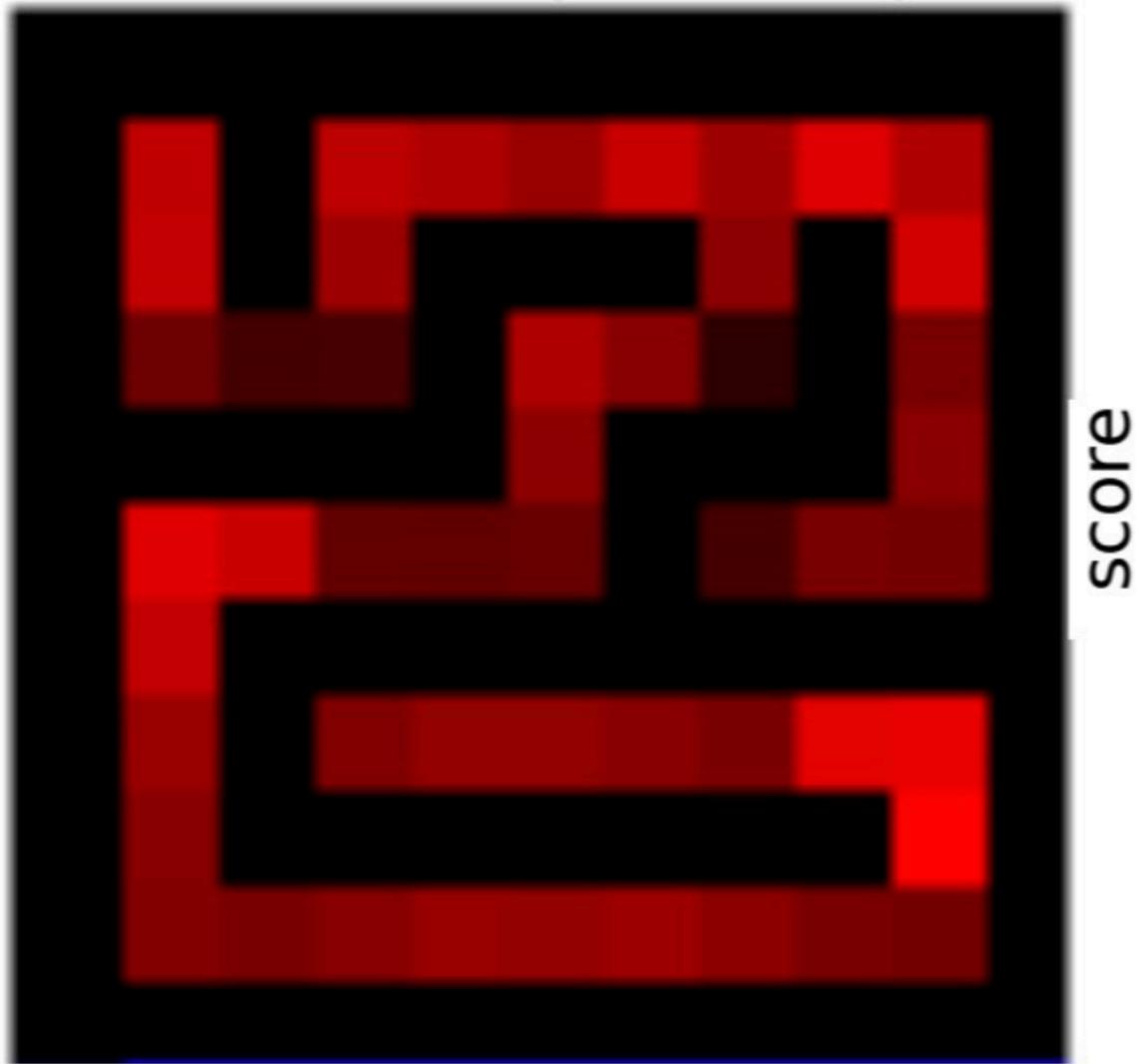


ATARI:
Amidar



2D maze

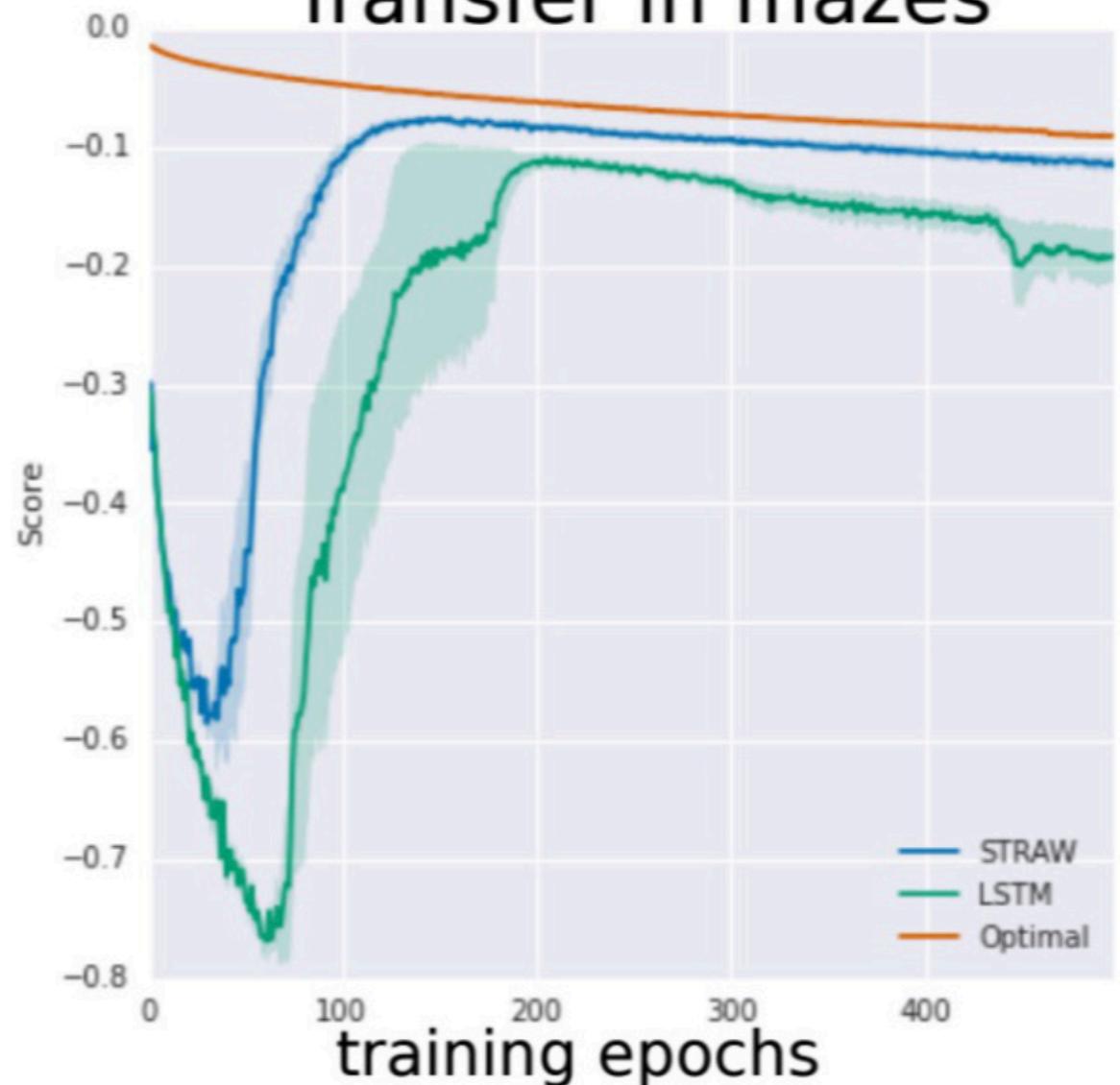
Maze replanning



STRAWe trained on an 11×11 random maze environment and then tested in a novel maze using varying start and goal locations.

The red intensity indicates how often the agent re-planned at a given location normalized by number of visits.

Transfer in mazes



Shifting to a 15×15 maze, STRAW is compared to an LSTM baseline and the optimal path given by the Dijkstra algorithm. As training epochs (2 million training steps) progress, the goal is move progressively further away from the start point. LSTM starts to show a strong drop-off compared to optimal policy, but STRAW continues to perform well.

[Vezhnevets et. al, 2016] Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, and Koray Kavukcuoglu. Strategic Attentive Writer for Learning Macro-Actions. In *NIPS*, 2016. (<https://arxiv.org/pdf/1606.04695.pdf>)

[Gregor et. al, 2015] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015. (<https://arxiv.org/pdf/1502.04623.pdf>)