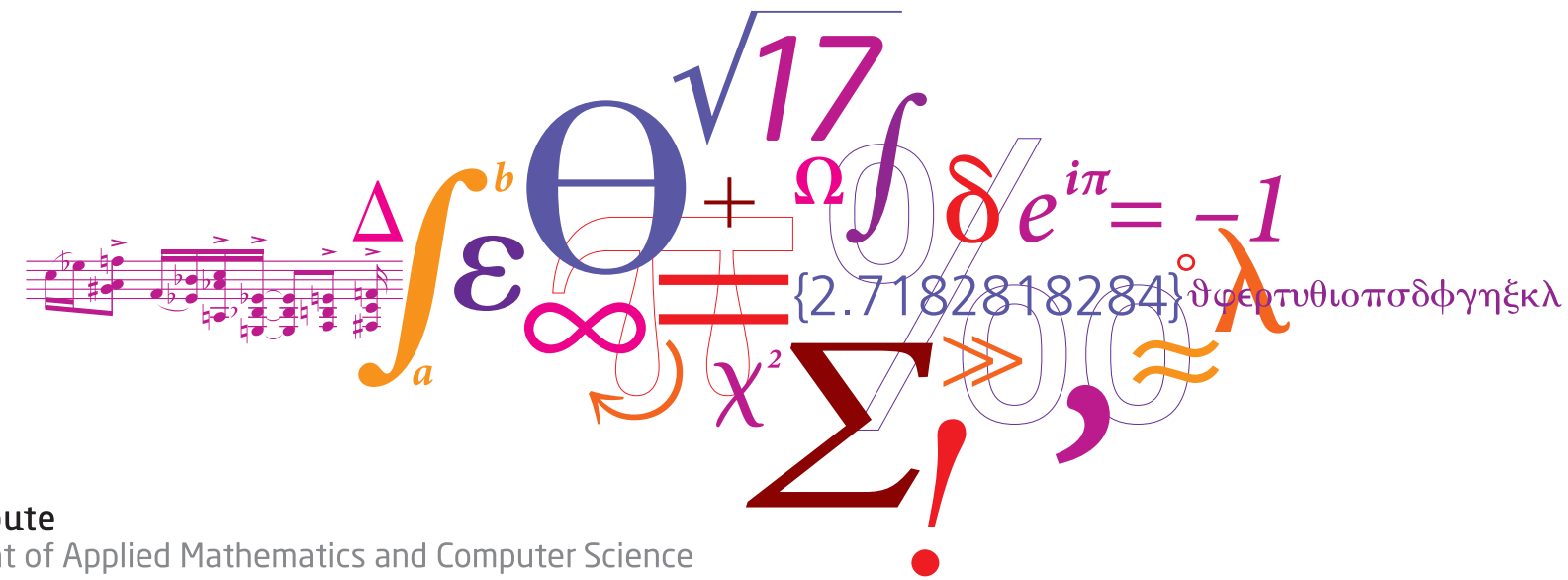
Learning to Plan from Raw Data in Grid-based Games

Andrea Dittadi and Thomas Bolander,
DTU Compute, Technical University of Denmark
Joint work with Ole Winther, DTU Compute



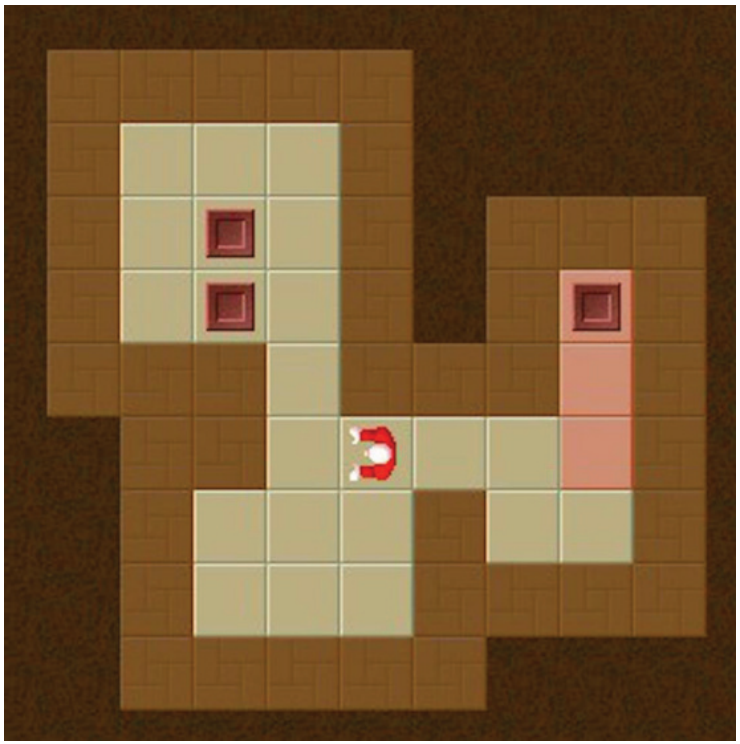
DTU Compute
Department of Applied Mathematics and Computer Science

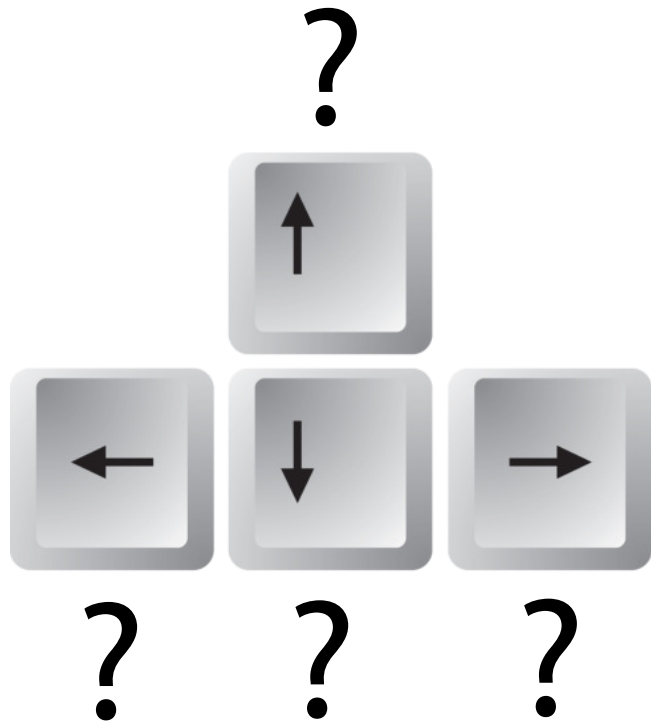
Why learning in planning?

When humans solve planning problems, it is not always the case that:

1. We know exactly what the **precondition** and **effects** are of all our **atomic actions**.
2. We have a **built-in heuristics** to guide our search for a goal.

Insofar as AI is about making computers do human-like things, we should try to make them work more on human-like premises, not give them an unrealistic head start.





Levels of AI planning

1 **Search problems**: Given initial state, actions, transition model, goal, and heuristics. **Weakness**: Every new problem requires new transition model and new hand-crafted heuristics. Solved by:

1.1 **Automated planning**: Actions/transition model given in PDDL (or similar). Heuristics automatically inferred. **Weakness**: Action schemas have to be hand-crafted. Solved by:

1.1.1 **Planning + learning of action models**: Level 1.1 + learning PDDL action models from observing state transitions. **Weakness**: We need symbolic observations. Alternative:

1.2 **Search + learning implicit transition models**: Level 1 + learning implicit transition models from raw data using e.g. neural networks. **Weakness**: No automatic inference of heuristics. Solved by:

1.2.1 **Search + learning implicit transitions models + learning heuristics**: Level 1.2 + learning heuristics from raw observations. **Weakness**: Transfer learning, explainability.

Symbolic vs sub-symbolic AI

The symbolic paradigm (1950–): Simulates human symbolic, conscious reasoning. Search, planning, logical reasoning. **Ex**: chess computer.

robust, predictable, explainable

strictly delimited abilities

flexible, learning

never 100% predictable/error-free

The sub-symbolic paradigm (1980–): Simulates the fundamental physical (neural) processes in the brain. Artificial neural networks. **Ex**: image recognition.

Google DeepMind's AlphaGo (2016) (and the movie by the same name)

CPH:DOX, 20 March 2018. Left to right: Fan Hui, European Go champion; Sebastian Risi, IT University of Denmark; me; Josh Rosen (producer of AlphaGo).

Google DeepMind playing Breakout

Combining reinforcement learning and neural networks. Reported in Nature vol 518, 26 February 2015.

`http://www2.compute.dtu.dk/~tobo/DeepMind.MP4`

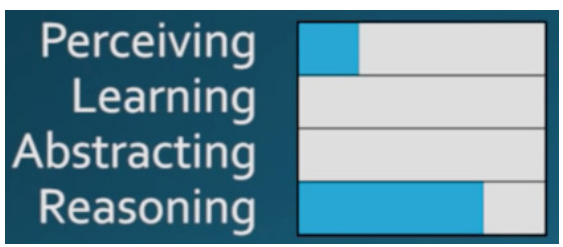
DeepMind playing Montezuma's Revenge

http://www2.compute.dtu.dk/~tobo/MontezumasRevenge.mov

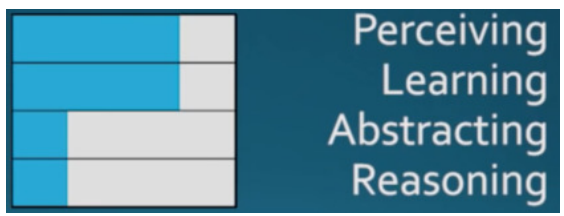
Three waves of AI

In February 2017, DARPA publishes the video "A DARPA Perspective on Artificial Intelligence" (https://www.youtube.com/watch?v=-O01G3tSYpU). Identifies three waves of AI, and relate to the DARPA challenge:

- *"The first wave of AI: Handcrafted knowledge"*. Essentially the symbolic paradigm.
- *"The second wave of AI: Statistical learning"*. Essentially the subsymbolic paradigm.
- *"The third wave of AI: Contextual adaptation"*. Essentially the combination of symbolic and subsymbolic approaches. Combine perception in neural networks with symbolic models for representing features, allowing **explanations** ("I thought it was a cat because it has fur and a short snout").



First wave



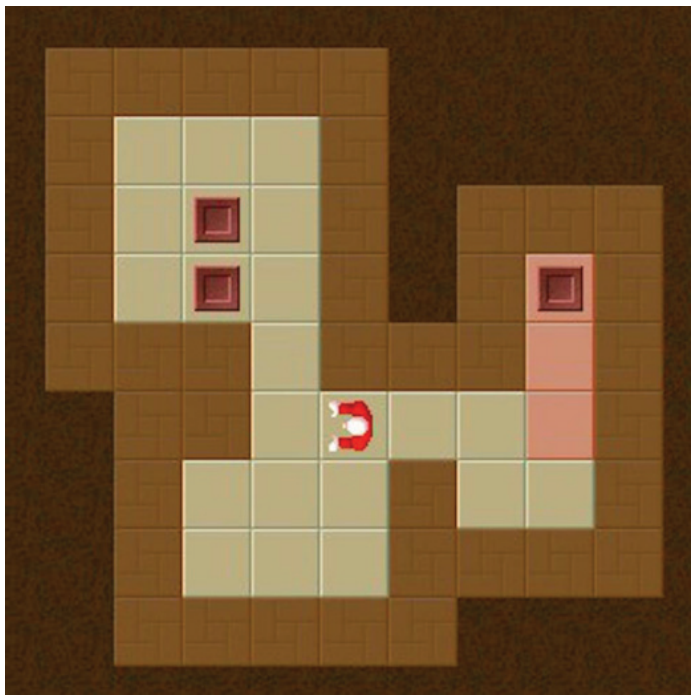
Second wave

Third wave

Motivation

Wish to create an artificial agent in a grid-based video game setting that:

- interacts with its environment, a fixed type of domain
- the domain can have several different instances (e.g. several different levels in a game)
- learns the environment dynamics from experience in the form of raw observations
- acts to achieve goals, and goals can differ between instances.



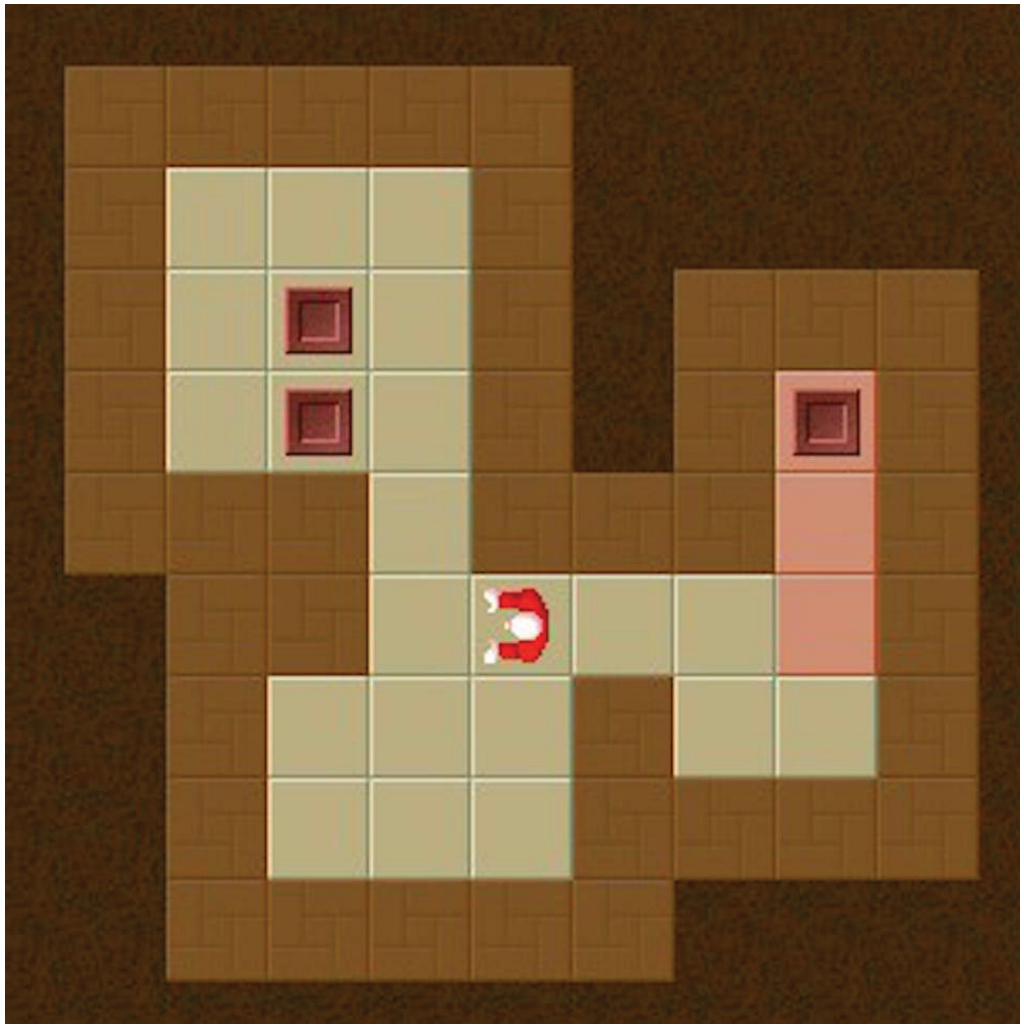
Possible approaches to solve our problem

- **Automated planning**. **Problem**: requires that the domain is already fully described in a logical formalism, like STRIPS below.

  $\textsc{Action}$: Move(player, from, to, dir)
  $\textsc{Prec}$: At(player, from) $\wedge$ Clear(to) $\wedge$ Move-Dir(from, to, dir)
  $\textsc{Effect}$: $\neg$At(player, from)$\wedge\neg$Clear(to)$\wedge$At(player, to)$\wedge$Clear(from)

  Unlike the problem a human novice is facing.
- **Action schema learning** in automated planning. **Problem**: requires observations to be symbolic. Still unlike the problem a human novice is facing.
- **Classical reinforcement learning** (model-free and model-based). **Problem**: 1) does not generalise to new instances of the domain, 2) does not generalise to new goals. Still not comparable to humans.
- **Our solution**: learn implicit (sub-symbolic) action models from raw observations, and then do search/planning based on those learned action models.

Sokoban



Learning framework

Sokoban is an instance of a **planning problem**: tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, s_0)$ where

- $\mathcal{S}$, $\mathcal{A}$, $\mathcal{G}$ are finite sets of **states**, **actions**, and **goal states** ($\mathcal{G} \subseteq \mathcal{S}$).
- $s_0 \in \mathcal{S}$ is an **initial state**.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a **deterministic transition function**.
- $s' = \mathcal{T}(s, a)$ is defined for all s and a, with $s' = s$ if a is not **applicable** in s.
- $\mathcal{G}$ is given as a **goal test function** $g : \mathcal{S} \to \{0, 1\}$, $g(s) = \mathbf{1}_{\mathcal{G}}(s)$.

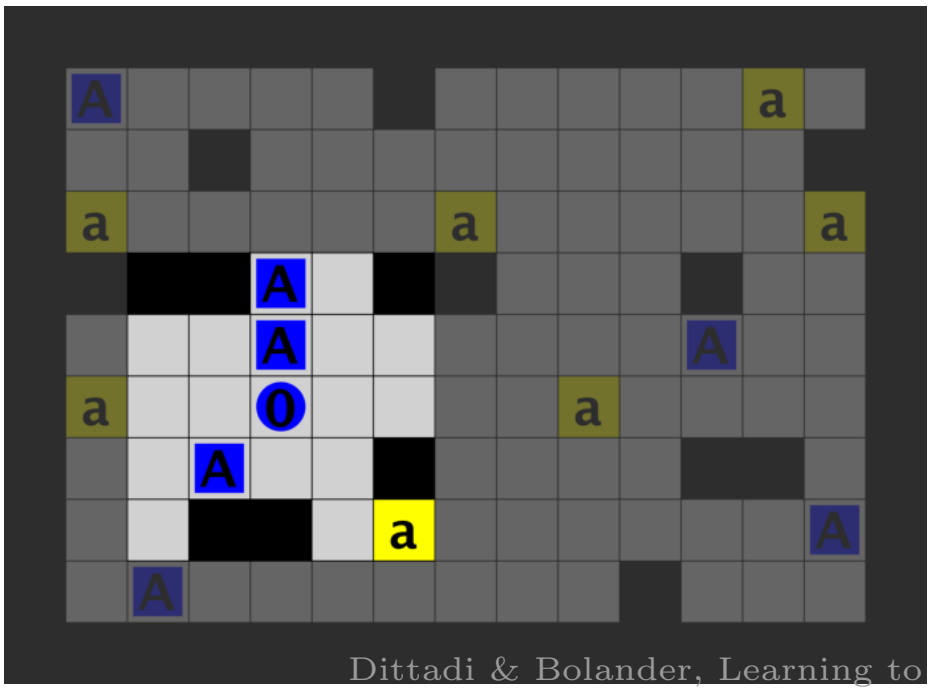
The states are fully observable, and each observation is a visual frame, i.e. a matrix of integers in the range $[0, 255]$ (8-bit grayscale representation of the individual pixels in the frame).

The agent is assumed to be aware of its own position in the environment.

For the noisy scenario, Gaussian independent additive noise.

Locality assumption: Action preconditions and effects are geometrically local

- Assume that the preconditions and effects of all actions are restricted to a local K-neighbourhood of the agent (a sub-grid of size $K \times K$ around the agent). K is either fixed or learned from observations.
- The K-neighbourhood is called the **local state** of the agent. We use $L^K_{i,j}(s)$ to denote the K-neighbourhood with center (i, j) of state s. When (i, j) is the position of the agent, we simply write $L^K(s)$. And when K is clear, $L(s)$.



If s denotes the state on the left, the highlighted local neighbourhood is $L^5_{7,5}(s) = L^5(s)$.

Environment prediction: learning the (local) dynamics of the environment

Let $\mathcal{S}_L$ be the set of local states (for a fixed K). The aim for the agent is to learn the **local transition function** $\mathcal{T}' : \mathcal{S}_L \times \mathcal{A} \to \mathcal{S}_L$ defined by

$$\mathcal{T}'(L^K_{i,j}(s), a) = L^K_{i,j}(\mathcal{T}(s, a)).$$

Note: This function is well-defined due to the locality assumption.

It maps a local neighbourhood and an action to the modified local neighbourhood after action execution.

When the locality assumption holds for a sufficiently small K, this is better than traditional reinforcement learning (RL):

- Generalises to new domain instances like ordinary action schemas (RL methods rely on the whole state space so in a new instance they have to re-learn).
- Easier to learn small area.

How the agent does environment prediction

The agent uses neural networks to learn an approximation $\hat{\mathcal{T}}'$ of the local transition function $\mathcal{T}'$:

$$\hat{\mathcal{T}}'(\Phi(L(s)), a; \boldsymbol{\theta}) \approx \Phi(\mathcal{T}'(L(s), a))$$

- Φ is a preprocessing map—the agent sees the world through this function, e.g. bit-depth reduction, downscaling, grayscale conversion.
- $\boldsymbol{\theta}$ is a vector of parameters for the neural networks (weights, biases).

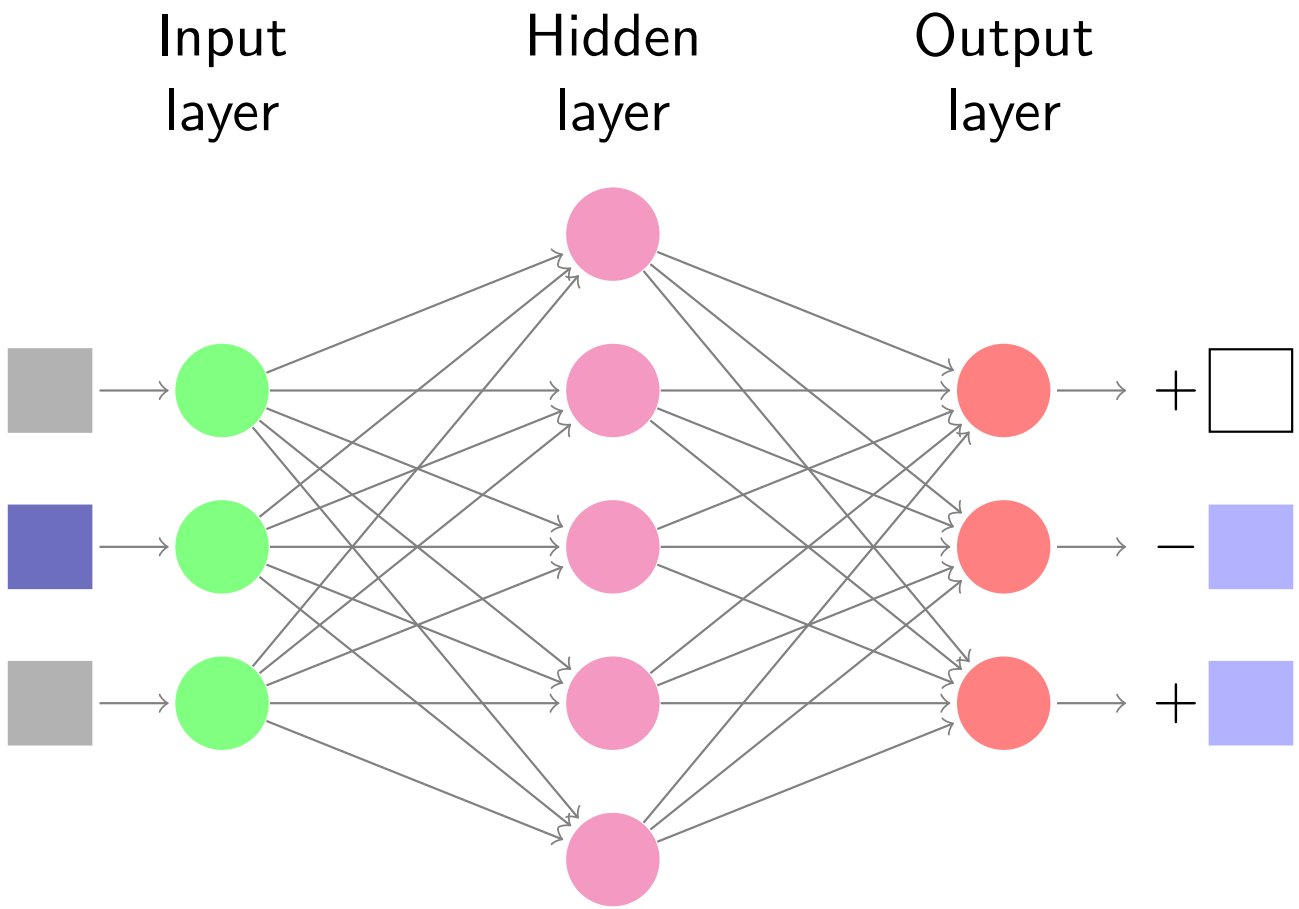
More precisely, we define (omitting normalisation operators):

$$\hat{\mathcal{T}}'(\Phi(s_L), a; \boldsymbol{\theta}) = \mathrm{NN}_a(\Phi(s_L); \boldsymbol{\theta}_a) + \Phi(s_L),$$

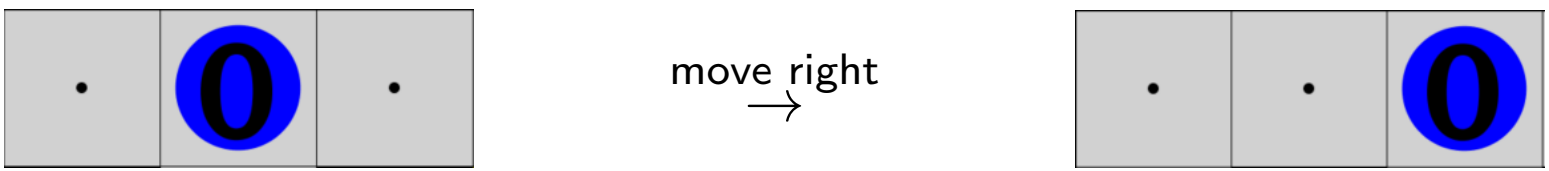
where NN_a is a neural network for learning action a (one network for each action).

Neural network architecture: K^2 inputs, 4 fully connected layers of size 64, 128, 128, 64, with leaky ReLU activations, and a linear fully connected output layer with K^2 outputs.

Illustration of environment prediction



Move right action network

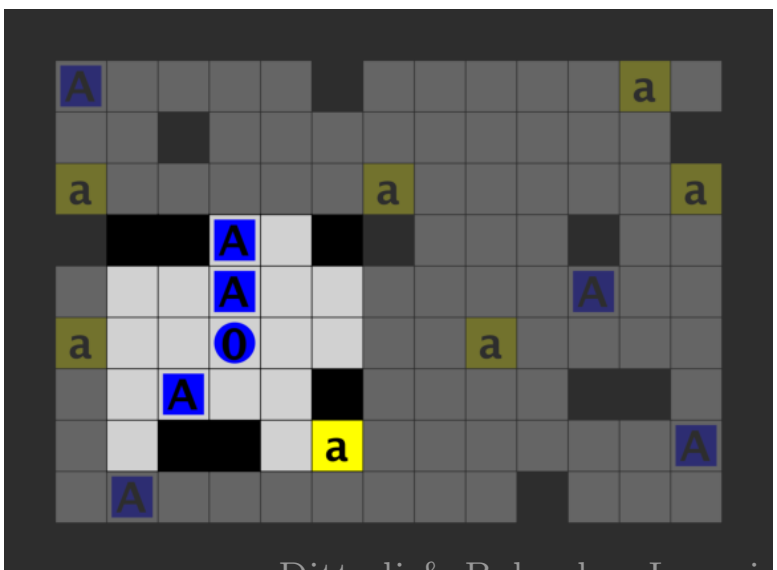


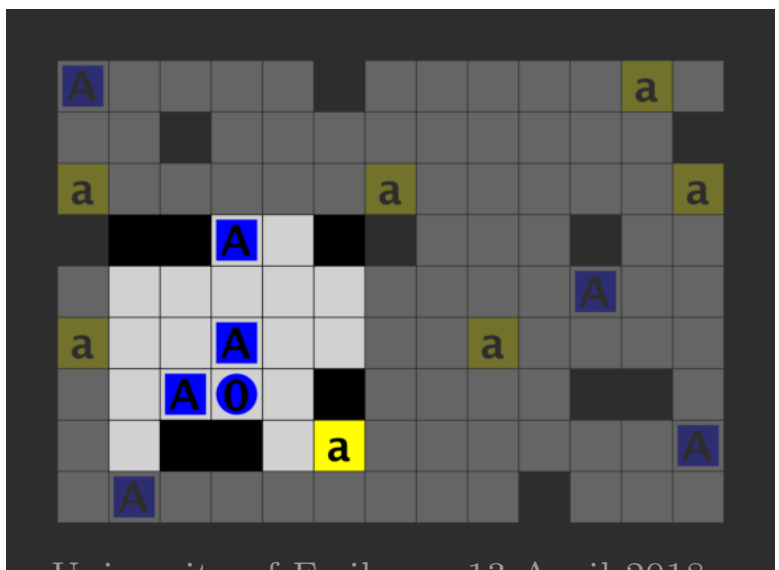
Training

- **Observe** local transition for action a in state s, that is, observe transition pair $(s, s') = (\Phi(L(s)), \Phi(\mathcal{T}'(L(s), a)))$.
- From $\Phi(L(s))$ the neural network has to **predict** the difference

$$s' - s = \Phi(\mathcal{T}'(L(s), a)) - \Phi(L(s)).$$

  Note relation to learning action schema effects.
- Minimize square **loss** by stochastic gradient descent, using Uniform Experience Replay (UER) (randomly sample a minibatch from large buffer of observations) and/or Prioritized Experience Replay (PER) (recently mispredicted transitions).





Planning

On top of the neural networks for approximate prediction of state transitions, we do traditional tree/graph search (e.g. BFS, DFS, A*):

- **Initial state** is current observed state.
- **Goal states** are given by goal test function.
- **Node expansion** using $\hat{\mathcal{T}}'$ (approximate state transition function given by the neural network predictors).

Key idea: compact action descriptions for planning, but choosing implicit representations (by NNs) instead of explicit (e.g. STRIPS).

If no plan is found (e.g. because environment model is too inaccurate), return to exploratory behaviour...

Exploration strategies

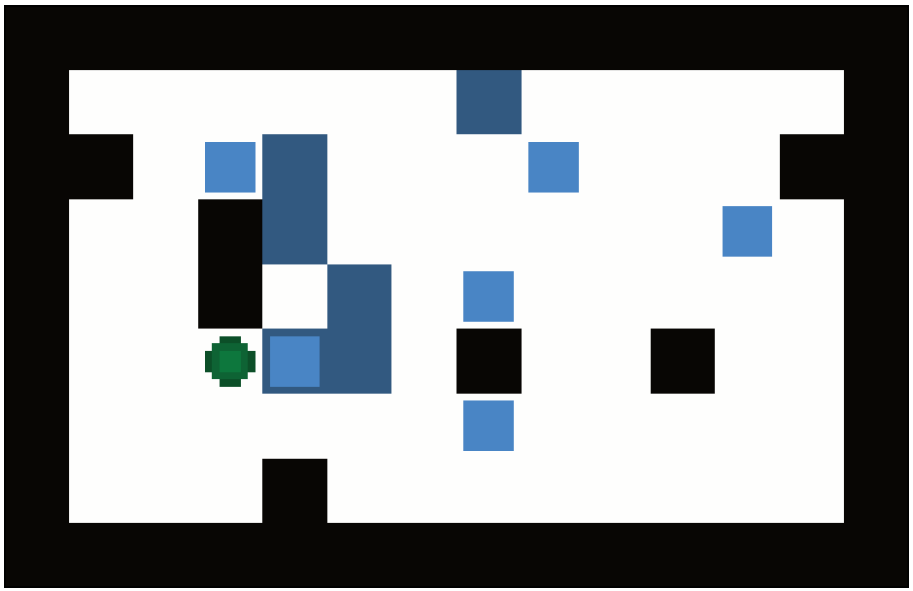
Exploration strategy: The agent's strategy for achieving data for learning (which actions to choose).

Better exploration $\rightarrow$ better training set $\rightarrow$ better learning

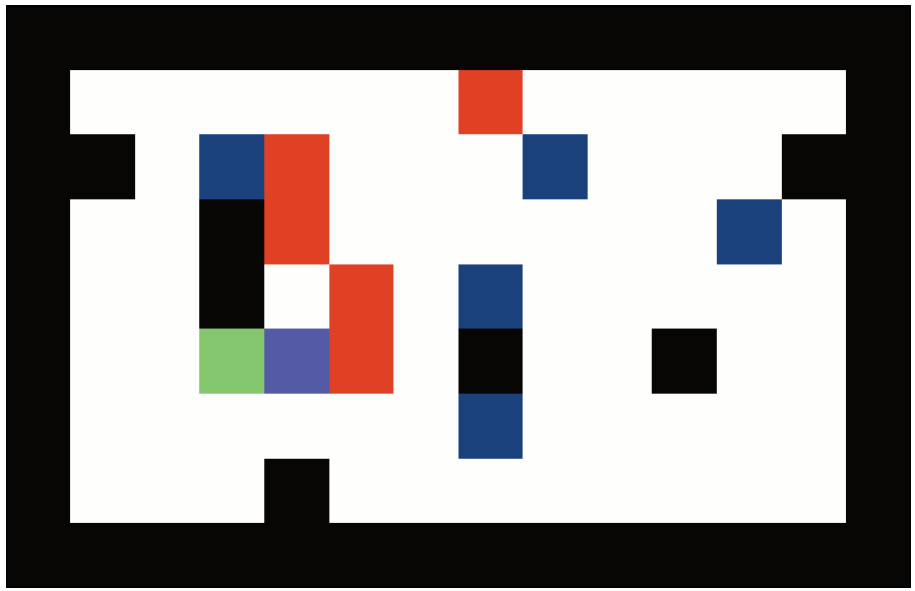
- **One-step exploration policies**: In each step, choose an action according to a (probabilistic) action selection policy. We consider two such policies π:
  - **Random exploration**: $\pi(a \mid s) = 1/|\mathcal{A}|$.
  - **Count-based exploration (preferring novelty)**: $\pi(a \mid s)$ is an action for which the number of explorations of a in the local state $L^K_{i,j}(s)$ is minimal.
- **Planning-based exploration (planning for novelty)**: Use planning to reach a state s, so that for some action a, the number of explorations of $(L^K_{i,j}(s), a)$ is minimal among all (local state, action) pairs. Execute a when s has been reached. (Essentially: Count-based + planning.)

Results – Sokoban observations

What the agent observes (two alternatives) and how it preprocesses it.



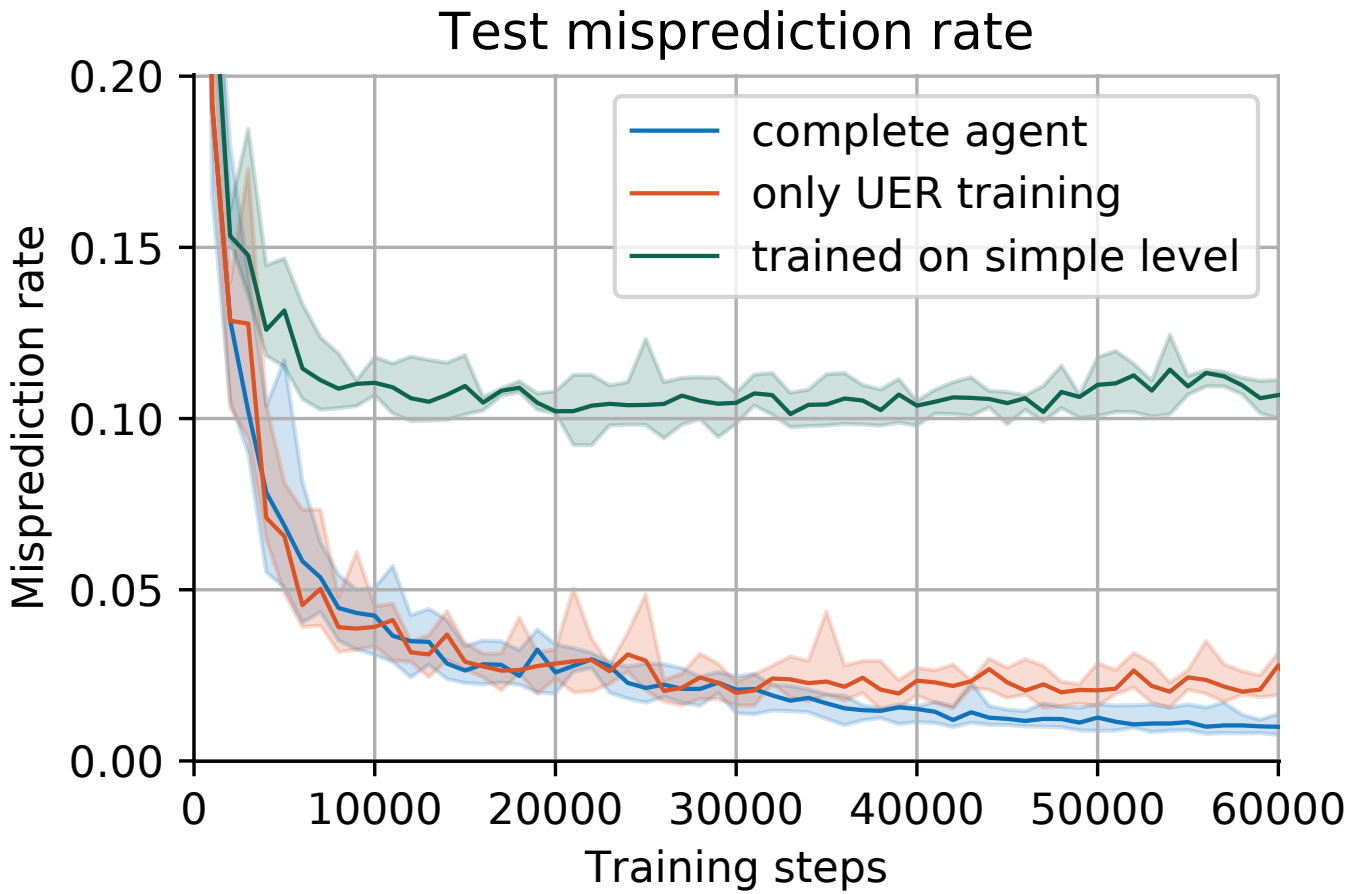
Pixel observations
(9x9 pixels per cell)



Low-dimensional observations
(one pixel per cell)

- The agent knows the frame size and the board size
- Observations are converted to grayscale and quantized. Pixel observations are also downscaled by a factor of 3 (pictures later).

Results – learning dynamics

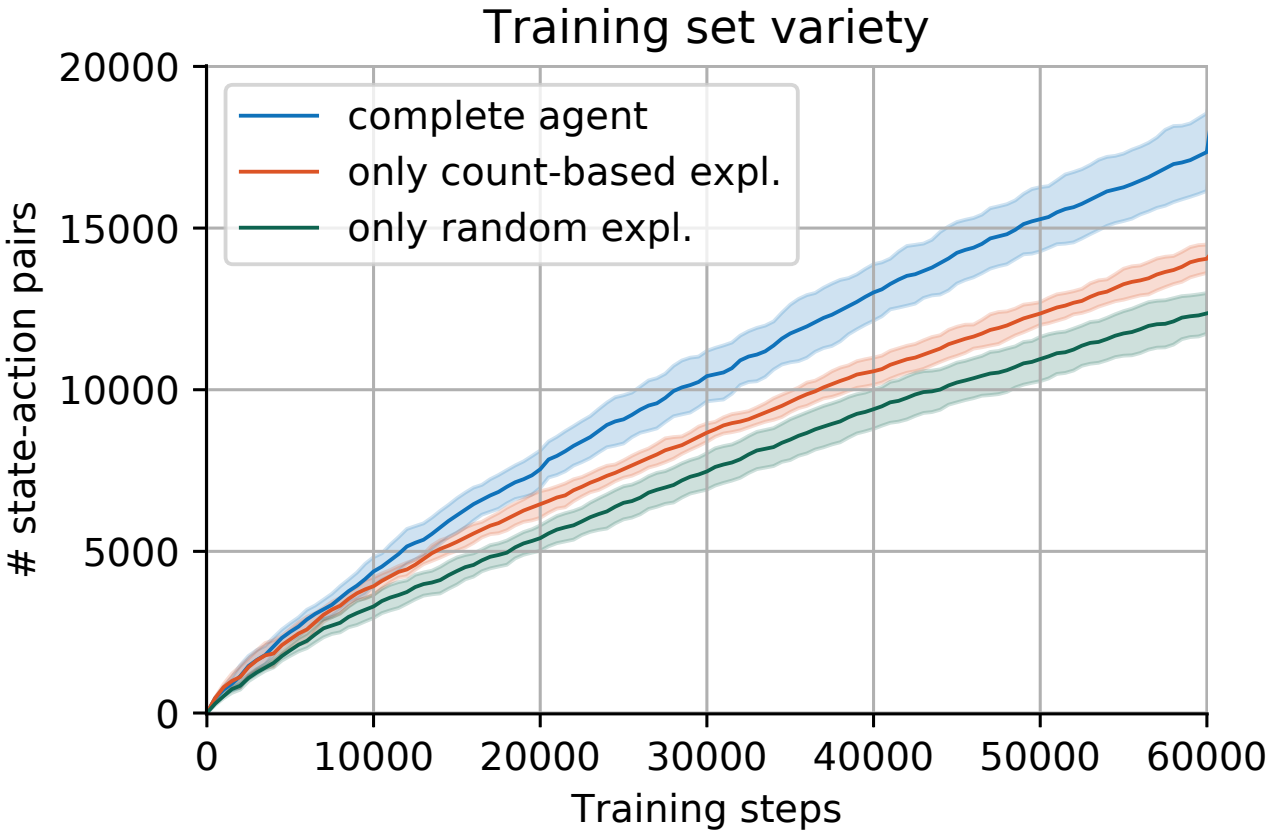


Misprediction rate on a test set of randomly generated local transitions.

Training on one level only: would be infeasible for RL/ML methods.

- Training level strongly affects performance
  - Complex and large level → diverse training set → better learning.
  - In this example, the simple level is a small level with 2 boxes only.
- The complete agent uses Prioritized Experience Replay (PER) and planning for exploration.

Results – exploration

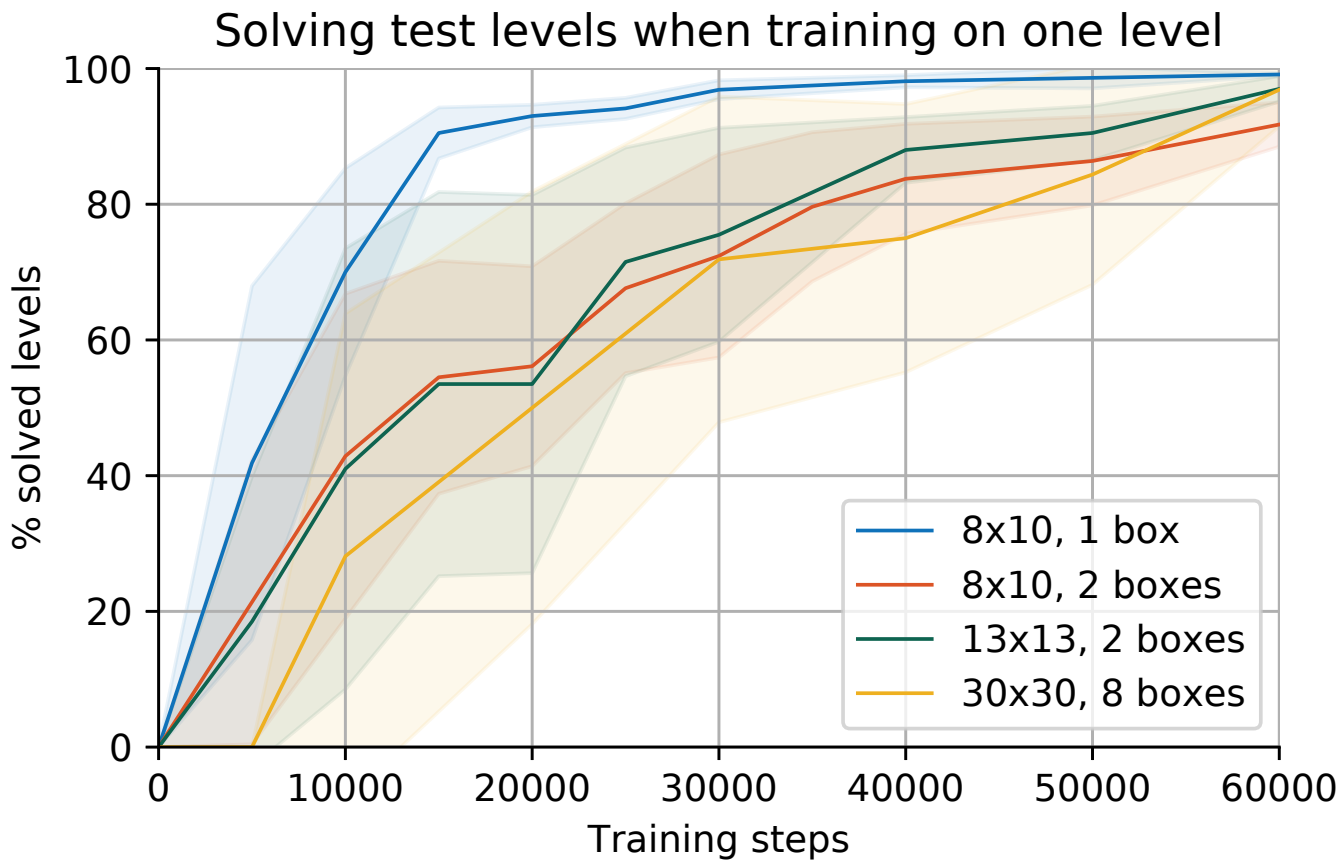


Number of distinct visited (state, action) pairs, while training on standard training level.

Blue line: one-step count-based exploration interleaved with planning for exploration.

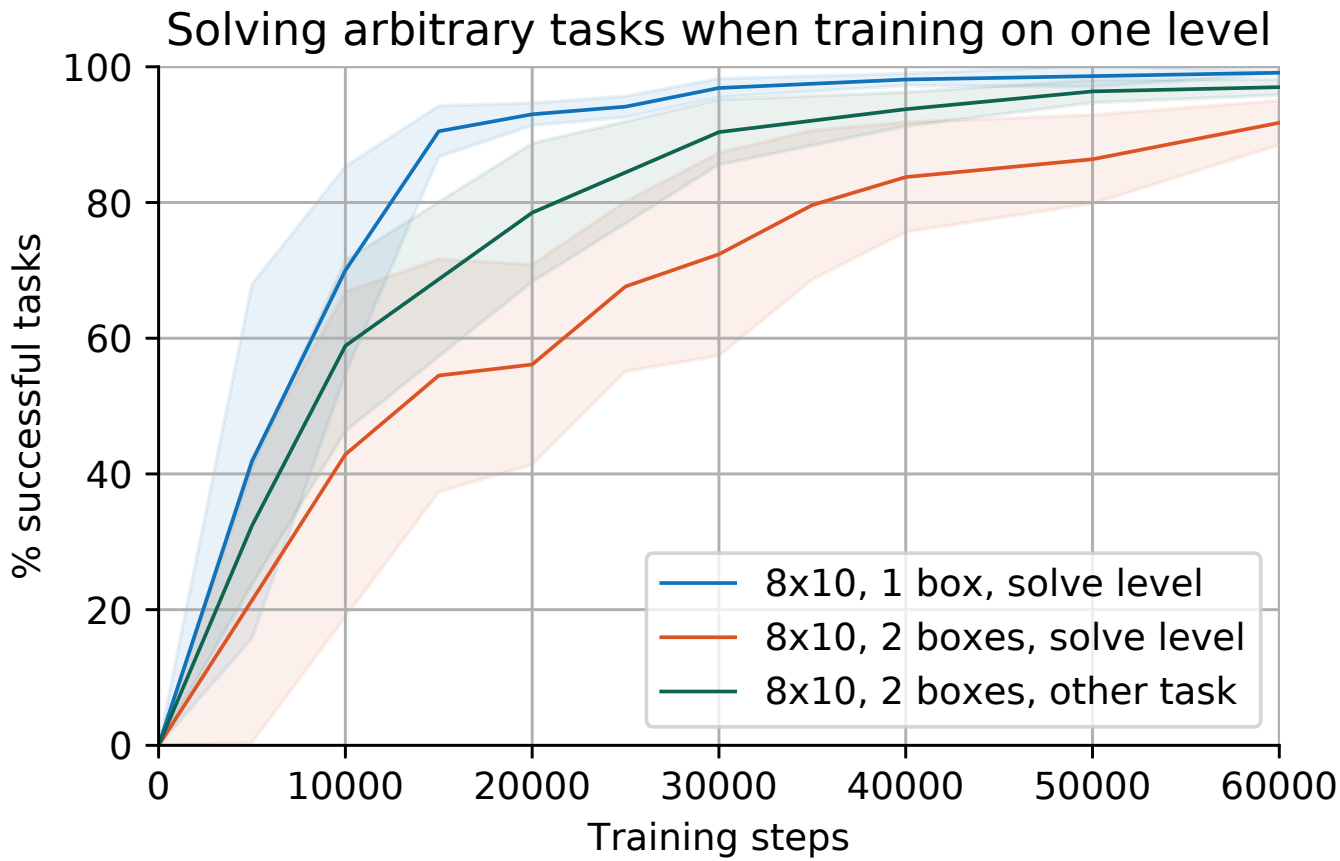
- More diverse experience → more learning potential.
- Best results: combination of one-step count-based exploration and planning for exploration.
  - Actively explore regions the agent would not explore with one-step count-based exploration alone, but when it doesn't know how to plan it can still explore in a smart way.

Results – solving levels



- Using BFS to search for goal states. On 30x30 levels we use A* with a simple domain-specific heuristics.
- Each line corresponds to a test set of randomly generated Sokoban levels.
- The only significant difference is when there is only 1 box (dynamics in the 1-box case are simpler, and learned more easily).

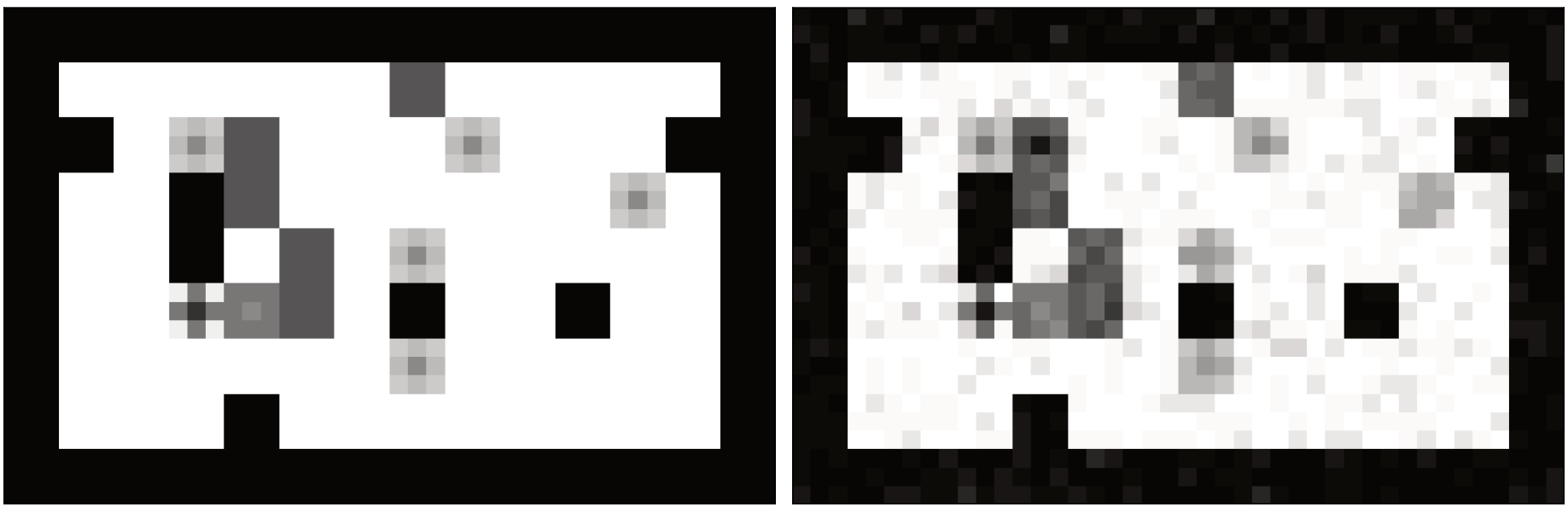
Results – solving new tasks



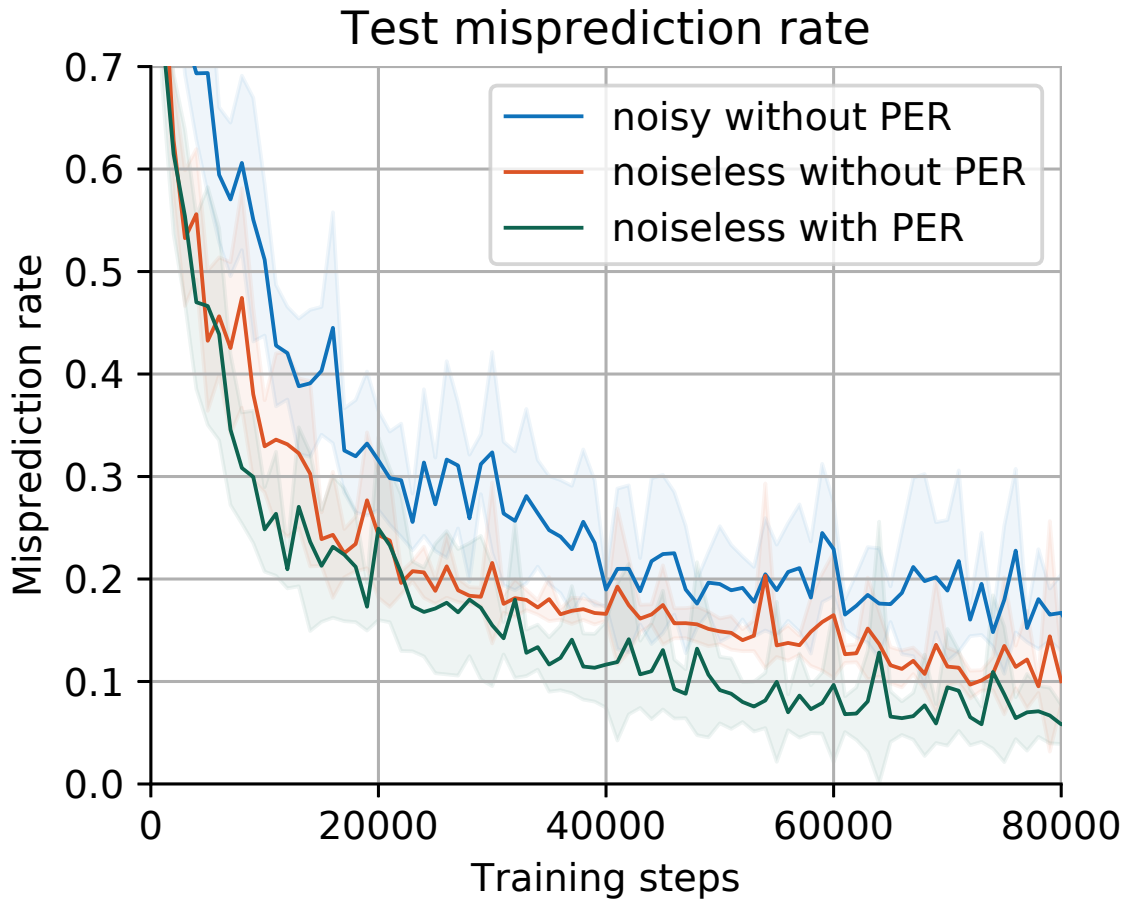
- The two *solve level* lines are as on the previous slides. The other task is to have at least one box next to each target.
- No additional training is needed for solving new tasks.
  - In RL a new policy would have to be learned from scratch from an updated reward signal.

Results – generalizing to pixels observations

- Generalisation of the learning setting.
- The pixel observations shown earlier are preprocessed by the "vision" map Φ.
- The agent learns from higher-dimensional observations shown here
  - noisy observations on the right (independent Gaussian)



Results – learning from pixels



- Noisy with Prioritized Experience Replay (PER) doesn't make sense.
- Performance degradation, but not bad for that level of noise.
- Would be difficult to achieve in most approaches to action schema learning.

Related work

- **Learning action schemas**
  [Amir, 2008, Walsh and Littman, 2008, Mourão et al., 2012]
  - efficient and flexible
  - but needs high-level logical observations.
- **Model-free reinforcement learning**
  [Mnih et al., 2015, Mnih et al., 2016]
  - learns from (high-dimensional) raw observations and a scalar reward signal
  - but cannot generalise to new domain instances
  - and has no planning capability.
- **Model-based reinforcement learning**
  [Tamar et al., 2016, Weber et al., 2017]
  - additionally learns environment model to do planning
  - but planning is typically in terms of faster convergence, the output is still an instance specific policy.
- Reinforcement learning methods are also inefficient: no assumption on environment structure, and possibly sparse rewards. Reasonable Sokoban results need 1 billion training samples [Weber et al., 2017].

Future work includes

- **Relax locality assumption**: more advanced learning of the relevant areas of preconditions and effects (compare Montezuma's Revenge).
- **Relax assumption of agent knowing its position**.
- **Improve learning from high-resolution pixel representations**: better machine learning models, e.g. convolutional neural networks.
- **Improve planning efficiency**: e.g. learning heuristics for informed state-space search.

APPENDIX

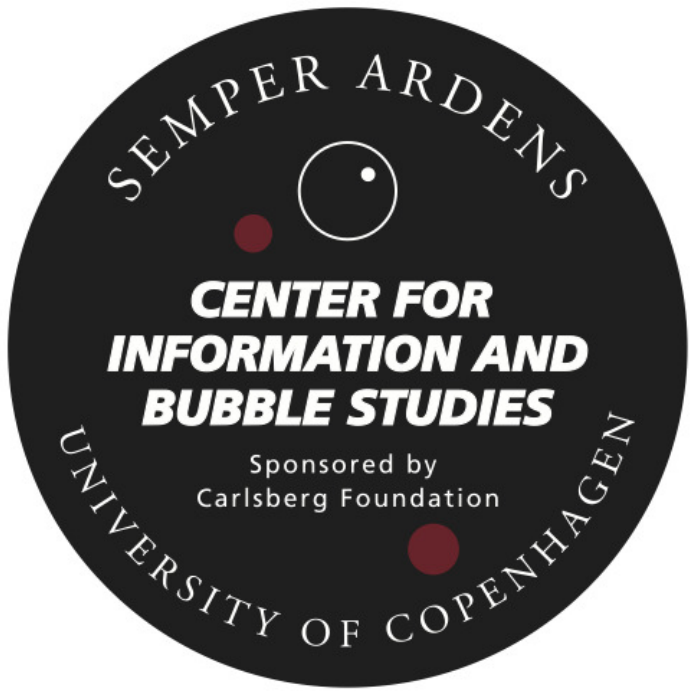
Reasoning about others in financial bubbles

Greater fools theory: Agent i is willing to buy an asset at a price p higher than its believed value, if i believes that the asset can later be resold for a price $> p$ (sold to "a greater fool").

$$\text{buy}_i(p) \leftrightarrow \left(B_i\left(v > p\right) \vee B_i \left(\bigvee_{j \in Agents} \bigvee_{p' > p} \text{buy}_j(p') \right) \right)$$

B_i: agent i believes that.

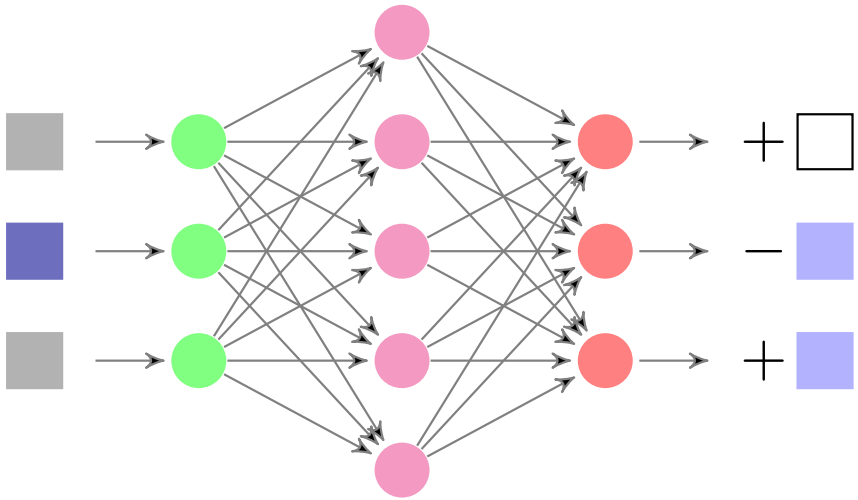
We study: Under which conditions does greater fool behaviour lead to financial bubble formation? What kind of agent types? What kind of reasoning abilities (levels of Theory of Mind)? What does it take to avoid those bubbles?



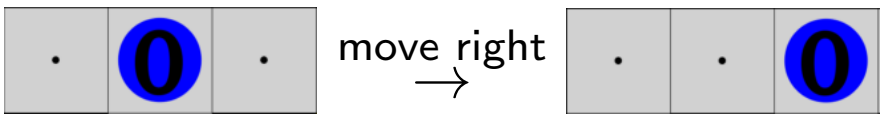
[Thomas Bolander and Hanna van Lee, in preparation]

Symbolic and Subsymbolic Action learning

Subsymbolic action learning



Move right action network



[Andrea Dittadi, Thomas Bolander & Ole Winther, under submission]

Symbolic action learning

ACTION : MoveRight(agt, from, to)

PRECONDITION :
$At(agt, from) \wedge Clear(to) \wedge RightOf(from, to)$

EFFECT :
$\neg At(player, from) \wedge At(player, to) \wedge \neg Clear(to) \wedge Clear(from)$

Currently extended to multi-agent learning under partial observability: learning actions and how others see them.

Joint work with Nina Gierasimczuk and Andrés Libermann.

Goal recognition

Planning-based goal recognition: Infer most probable goal from observing action sequence and inferring cost of achieving different goals.

By Bayes rule, the chain rule and the law of total probability:

$$\begin{aligned} P(g \mid a_1 \cdots a_n, s_0) &= \frac{P(a_1 \cdots a_n | s_0, g) P(g)}{\sum_{g' \in \text{Goals}} P(a_1 \cdots a_n | s_0, g') P(g')} \\ &= \frac{P(g) \prod_{j=1}^{n} P(a_j | T(s_0, a_1 \cdots a_{j-1}), g)}{\sum_{g' \in \text{Goals}} P(g') \prod_{j=1}^{n} P(a_j | T(s_0, a_1 \cdots a_{j-1}), g)} \end{aligned}$$

Used in Smart Innovation project with the startup company Shade making intelligent lighting (2017–2018).

[Joint work with Andrea Dittadi. Planning-based goal recognition is based on Ramirez, 2011; Geffner, 2013.]

Human child, 18 months old

`http://www2.compute.dtu.dk/~tobo/children_cabinet.mpg`

The child is *not* given any instructions beforehand.

[Warneken 2006]

Modelling the cabinet example

Possible reasoning and acting by child:

- **Goal recognition**. Infer goal g of human: papers in cabinet.
- **Multi-agent planning**. Compute plan (c is child, a is adult): c:*go_to_cabinet*, c:*open_cabinet*, a:*place_papers*.
- **Verify that plan is implicitly coordinated** (using beliefs, B, instead of knowledge, K):

$$s_0 \models B_r(\langle c\text{:}go\_to\_cabinet\rangle\top \wedge [c\text{:}go\_to\_cabinet]$$
$$B_c(\langle c\text{:}open\_cabinet\rangle\top \wedge [c\text{:}open\_cabinet]$$
$$B_a(\langle a\text{:}place\_papers\rangle\top \wedge [a\text{:}place\_papers]g)))$$

- Execute c-steps in plan. Wait for a-steps. Replan if failure.

References I

Amir, E. (2008).

Learning Partially Observable Deterministic Action Models.

Journal of Artificial Intelligence Research (JAIR) *33*, 349–402.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K. (2016).

Asynchronous methods for deep reinforcement learning.

In International Conference on Machine Learning (ICML) pp. 1928–1937,.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D. (2015).

Human-level control through deep reinforcement learning.

Nature *518*, 529—-533.

Mourão, K., Zettlemoyer, L. S., Petrick, R. P. A. and Steedman, M. (2012).

Learning STRIPS Operators from Noisy and Incomplete Observations.

In Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012, (de Freitas, N. and Murphy, K. P., eds), pp. 614–623, AUAI Press.

Tamar, A., Wu, Y., Thomas, G., Levine, S. and Abbeel, P. (2016).

Value Iteration Networks.

In Advances in Neural Information Processing Systems (NIPS) pp. 2154–2162,.

Walsh, T. J. and Littman, M. L. (2008).

Efficient Learning of Action Schemas and Web-service Descriptions.

In Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2 AAAI'08 pp. 714–719, AAAI Press.

References II

Weber, T., Racanière, S., Reichert, D. P., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y. et al. (2017).
Imagination-Augmented Agents for Deep Reinforcement Learning.
In Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS).