The Predictron: End-to-end Learning and Planning

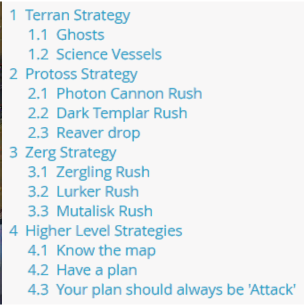
Yoonho Lee

Department of Computer Science and Engineering
Pohang University of Science and Technology

December 27, 2016

Hierarchical RL

motivation



Hierarchical RL

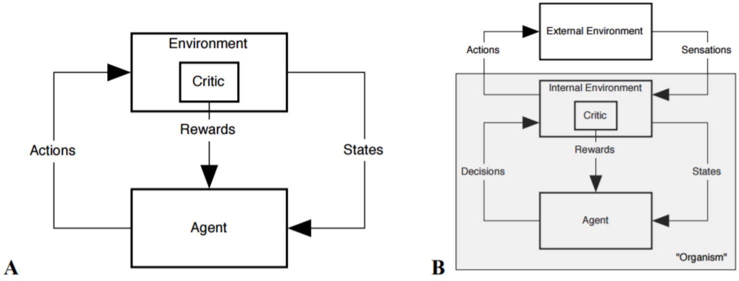


Figure 1: *Agent-Environment Interaction in RL.* **A**: *The usual view.* **B**: *An elaboration.*

Hierarchical RL

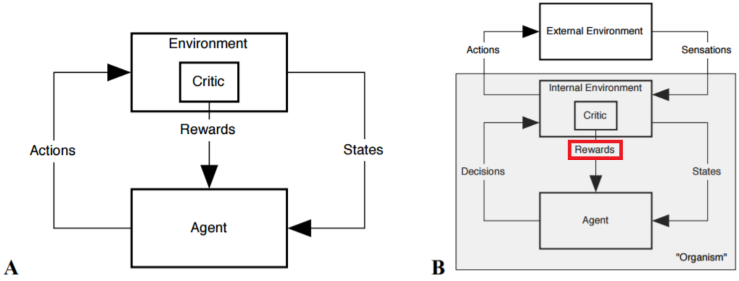


Figure 1: *Agent-Environment Interaction in RL.* **A**: *The usual view.* **B**: *An elaboration.*

Reward augmentation[1]

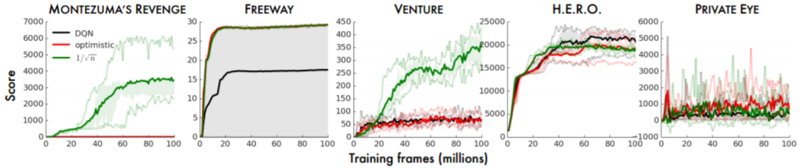


Figure 2: Average training score with and without exploration bonus or optimistic initialization in 5 Atari 2600 games. Shaded areas denote inter-quartile range, dotted lines show min/max scores.

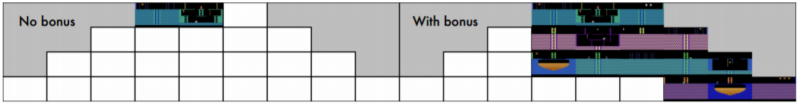


Figure 3: "Known world" of a DQN agent trained for 50 million frames with (**right**) and without (**left**) count-based exploration bonuses, in MONTEZUMA'S REVENGE.

[1] Bellemare et al. Unifying Count-Based Exploration and Intrinsic Motivation, NIPS 2016

Hierarchical RL

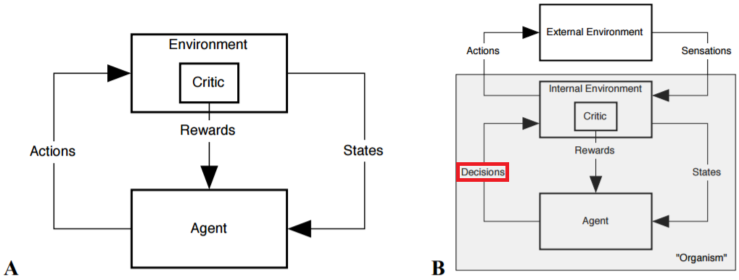
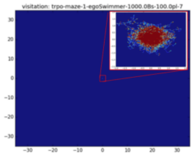
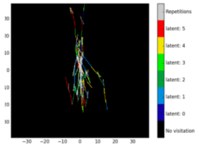


Figure 1: *Agent-Environment Interaction in RL.* **A**: *The usual view.* **B**: *An elaboration.*

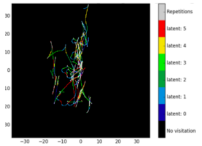
Hierarchical actions[2]



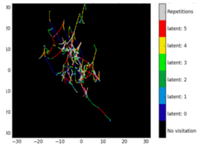
(a) Gaussian noise with covariance $\Sigma = I$



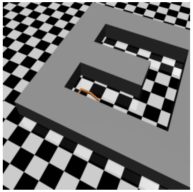
(b) Hierarchichy with Multi-policy



(c) Hierarchy with Bilinear-SNN(,$\alpha_H = 0$)

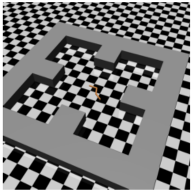


(d) Hierarchy with Bilinear-SNN, $\alpha_H = 0.01$

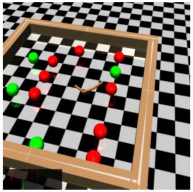


(a) Maze 0

(b) Maze 1



(c) Maze 2 or 3



(d) Food Gather

[2] Florensa et al. Stochastic Neural Networks for Hierarchical Reinforcement Learning, under review for ICLR 2017

Hierarchical RL

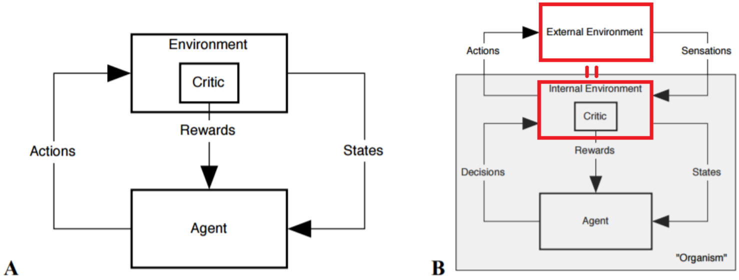
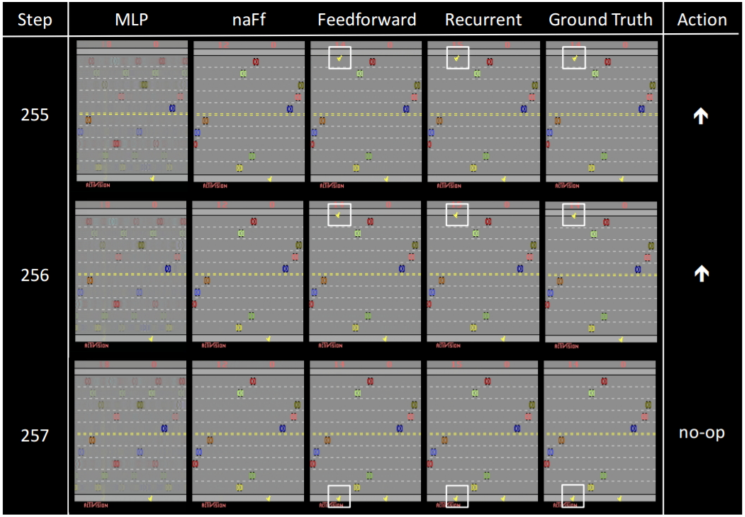


Figure 1: *Agent-Environment Interaction in RL.* **A**: *The usual view.* **B**: *An elaboration.*

Model-based video prediction[3]



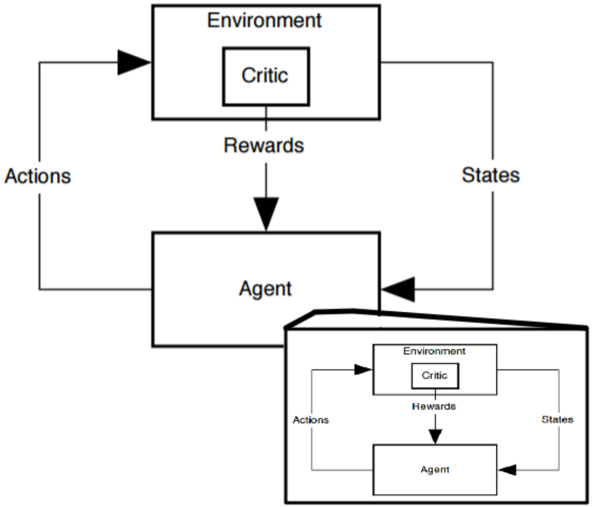
[3]Oh et al. Action-Conditional Video Prediction using Deep Networks in Atari Games, NIPS 2015

Value Iteration Networks

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, Pieter Abbeel
Best paper at NIPS 2016

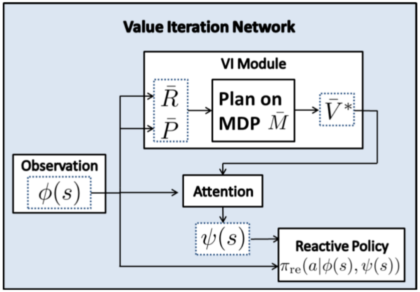
Value Iteration Networks

Motivation



Value Iteration Networks

Network diagram



Value Iteration Networks

Bellman Operator as a CNN forward pass

Bellman Operator

Every V converges to V^* under the Bellman Operator T defined as:

$$(TV)(s) = \max_{a \in A}\{R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V(s')\} \tag{1}$$

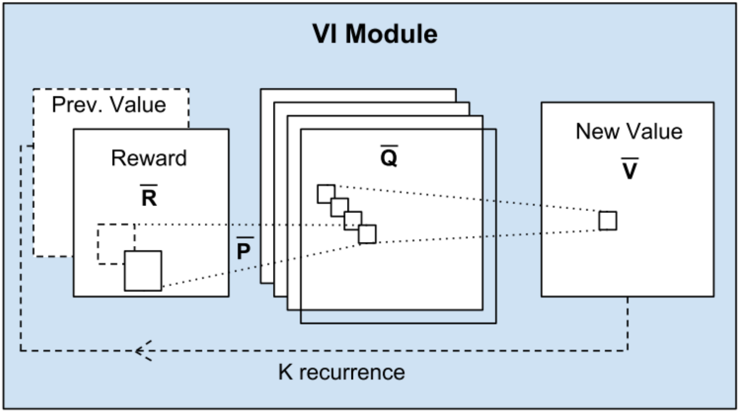
$$V^* = \lim_{n \to \infty} T^n V \quad \forall V \tag{2}$$

The Bellman Operator can be viewed as a CNN forward pass:

$$(T^*V)(s) = \max_{a \in A}\{R(s,a) + \gamma conv_{P(a)}(V(s))\}$$

Value Iteration Networks

VI Module



Value Iteration Networks

Results

Network	Train Error	Test Error
VIN	0.30	0.35
CNN	0.19	0.73

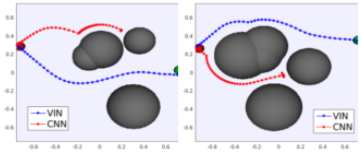


Figure 4: Continuous control domain. Left: average distance to goal on training and test domains for VIN and CNN policies. Right: trajectories predicted by VI and CNN on random test domains.

Value Iteration Networks

Summary

- Neural network architecture that plans using value iteration
- Assumes that the state is a sufficient statistic for the reward function
- The small MDP must have a finite state space
- Uses prior knowledge about the environment's structure

The Predictron: End-to-End Learning and Planning

David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul,
Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert,
Neil Rabinowitz, Andre Barreto, Thomas Degris

Under review for ICLR 2017

Predictron

$$\begin{tikzcd}
\vdots \\
\mathbf{r}^2, \mathbf{s}^2, \gamma^2 \arrow[u, "m"] \arrow[r, "v"'] & \mathbf{v}^2 \\
\mathbf{r}^1, \mathbf{s}^1, \gamma^1 \arrow[u, "m"] \arrow[r, "v"'] & \mathbf{v}^1 \\
\mathbf{s}^0 \arrow[u, "m"] \arrow[r, "v"'] & \mathbf{v}^0 \\
s \arrow[u, "f"]
\end{tikzcd}$$

(a)

$$\vdots$$

$$\mathbf{g}^2 = \mathbf{r}^1 + \gamma^1 \mathbf{r}^2 + \gamma^1 \gamma^2 \mathbf{v}^2$$

$$\mathbf{g}^1 = \mathbf{r}^1 + \gamma^1 \mathbf{v}^1$$

$$\mathbf{g}^0 = \mathbf{v}^0$$

(b)

$$\begin{tikzcd}
\vdots \\
\mathbb{E}^m\left[\mathbf{g}^2\right] \arrow[u, equal] \\
\mathbb{E}^m\left[\mathbf{g}^1\right] \arrow[u, equal] \\
\mathbb{E}^m\left[\mathbf{g}^0\right] \arrow[u, equal] \\
\mathbb{E}_p\left[\mathbf{g}_t\right] \arrow[u, equal]
\end{tikzcd}$$

(c)

Predictron

$$\begin{tikzcd}
\vdots & \\
\mathbf{r}^2, \mathbf{s}^2, \gamma^2 \arrow[u, "m"] \arrow[r, "v"'] & \mathbf{v}^2 \\
\mathbf{r}^1, \mathbf{s}^1, \gamma^1 \arrow[u, "m"] \arrow[r, "v"'] & \mathbf{v}^1 \\
\mathbf{s}^0 \arrow[u, "m"] \arrow[r, "v"'] & \mathbf{v}^0 \\
s \arrow[u, "f"] &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
\vdots & & & \vdots \arrow[dd, dotted] \\
& & \boldsymbol{\lambda}^2 & \\
\mathbf{r}^2, \mathbf{s}^2, \gamma^2 \arrow[uu, "m"] \arrow[urr, dotted, bend left] \arrow[r, "v"'] & \mathbf{v}^2 \arrow[rr, dotted, "\mathbf{1}-\boldsymbol{\lambda}^2"] & & \mathbf{g}^{2,\lambda} \arrow[dd, dotted] \\
& & \boldsymbol{\lambda}^1 & \\
\mathbf{r}^1, \mathbf{s}^1, \gamma^1 \arrow[uu, "m"] \arrow[urr, dotted, bend left] \arrow[r, "v"'] & \mathbf{v}^1 \arrow[rr, dotted, "\mathbf{1}-\boldsymbol{\lambda}^1"] & & \mathbf{g}^{1,\lambda} \arrow[dd, dotted] \\
& & \boldsymbol{\lambda}^0 & \\
\mathbf{s}^0 \arrow[uu, "m"] \arrow[urr, dotted, bend left] \arrow[r, "v"'] & \mathbf{v}^0 \arrow[rr, dotted, "\mathbf{1}-\boldsymbol{\lambda}^0"] & & \mathbf{g}^{0,\lambda} \arrow[d, dotted] \\
\mathbf{r}, s, \gamma \arrow[u, "f"] & & & \mathbf{g}^{\lambda}
\end{tikzcd}$$

(d)

Predictron

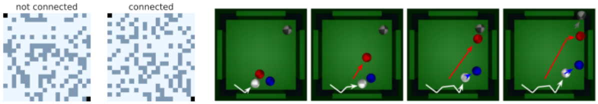


Figure 2: **Left:** Two sample mazes from the random-maze domain. Light blue cells are empty, darker blue cells contain a wall. One maze is connected from top-left to bottom-right (indicated in black), the other is not. **Right:** An example trajectory in the pool domain (before downsampling). It was selected by maximising the prediction of pocketing balls, using the predictron.

Predictron

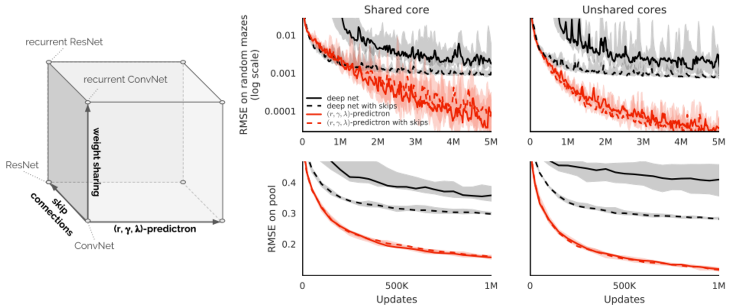


Figure 4: **Comparing predictron to baselines.** Aggregated prediction errors on random mazes (top) and pool (bottom) over all predictions for the eight architectures corresponding to the cube on the left. Each line is the median of RMSE over five seeds; shaded regions encompass all seeds. The full (r, γ, λ)-predictron (**red**), consistently outperformed conventional deep network architectures (**black**), with and without skips or and with and without weight sharing.

Predictron

$$\begin{tikzcd}
\vdots & & & \vdots \arrow[dd, dotted] \\
 & & \boldsymbol{\lambda}^2 & \\
\mathbf{r}^2, \mathbf{s}^2, \gamma^2 \arrow[uu, "m"] \arrow[r, "v"'] \arrow[urr, dotted, bend left] & \mathbf{v}^2 \arrow[rr, dotted, "\mathbf{1}-\boldsymbol{\lambda}^2"] & & \mathbf{g}^{2,\lambda} \arrow[dd, dotted] \\
 & & \boldsymbol{\lambda}^1 & \\
\mathbf{r}^1, \mathbf{s}^1, \gamma^1 \arrow[uu, "m"] \arrow[r, "v"'] \arrow[urr, dotted, bend left] & \mathbf{v}^1 \arrow[rr, dotted, "\mathbf{1}-\boldsymbol{\lambda}^1"] & & \mathbf{g}^{1,\lambda} \arrow[dd, dotted] \\
 & & \boldsymbol{\lambda}^0 & \\
\mathbf{s}^0 \arrow[uu, "m"] \arrow[r, "v"'] \arrow[urr, dotted, bend left] & \mathbf{v}^0 \arrow[rr, dotted, "\mathbf{1}-\boldsymbol{\lambda}^0"] & & \mathbf{g}^{0,\lambda} \arrow[d, dotted] \\
\mathbf{r}, s, \gamma \arrow[u, "f"] & & & \mathbf{g}^{\lambda}
\end{tikzcd}$$

(d)

$$\vdots$$

$$\mathbf{g}^2 = \mathbf{r}^1 + \gamma^1 \mathbf{r}^2 + \gamma^1 \gamma^2 \mathbf{v}^2$$

$$\mathbf{g}^1 = \mathbf{r}^1 + \gamma^1 \mathbf{v}^1$$

$$\mathbf{g}^0 = \mathbf{v}^0$$

(b)

Predictron

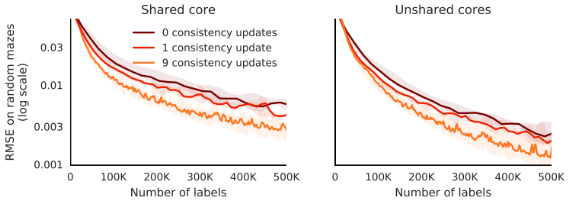


Figure 5: **Semi-supervised learning.** Prediction errors of the (r, γ, λ)-predictrons (shared core, no skips) using 0, 1, or 9 consistency updates for every update with labelled data, plotted as function of the amount of labelled data consumed. Early learning performance improved with more consistency updates.

Predictron

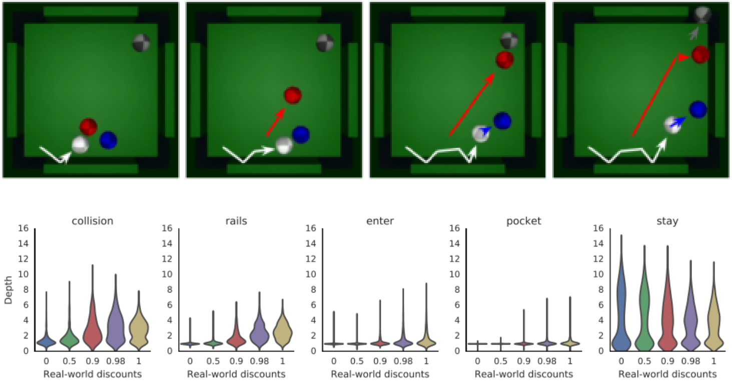


Figure 6: **Thinking depth.** Distributions of thinking depth on pool for different types of predictions and for different real-world discounts. Depth is defined as $\boldsymbol{\lambda}^0(1 + \boldsymbol{\gamma}^1\boldsymbol{\lambda}^1(1 + \boldsymbol{\gamma}^2\boldsymbol{\lambda}^2(1 + \ldots)))$.

Predictron

Summary

- End-to-end simulation of an MRP
- Works with arbitrary state space for small MRP
- Small MRP has a non-interpretable state space
- Designed for RL, but does not take actions into account

Summary

- Hierarchical Reinforcement Learning attempts to identify crucial decisions
- Agents can now use NN-based planning for better decision making

- Research directions
  - Theoretical bounds for optimal policy of a smaller MDP
  - Learning a smaller MDP with abstract actions
  - End-to-end planning based Q network or policy network

Thank You

Value Iteration Networks
The Predictron: End-to-End Learning and Planning