Reinforcement learning

Bonus I

Intrinsic motivation in RL

Yandex
Data Factory

LAMBDA

British Hedgehog
Preservation Society

Summary of RL course so far...

Yes we can!

Can solve RL for

- Finite/Infinite MDP
- Full or partial observability
- Large or continuous state space
- Small OR structured OR continuous action space (later)

Given

- Frequent rewards
- Short delay between reward and its cause
- Efficient exploration

Real world

Can solve RL for

- Finite/Infinite MDP
- Full or partial observability
- Large or continuous state space
- Small OR structured OR continuous action space (later)

Given

- ~~Frequent rewards~~
- ~~Short delay~~
- Efficient exploration

You're graded once in weeks

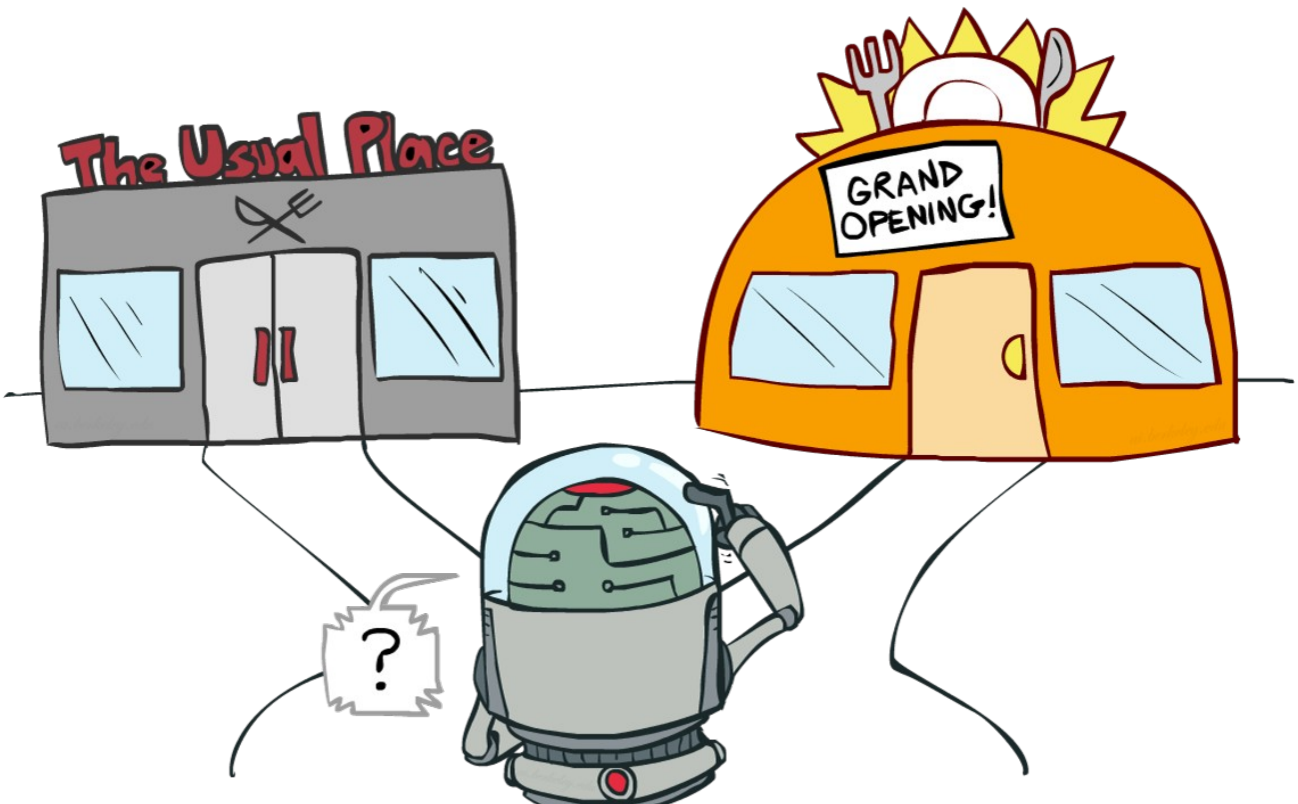
… for stuff you did months ago

because we suck at
checking up your homeworks

Exploration Vs Exploitation

Balance between using what you learned and trying to find something even better

How did we do that before?



Exploration strategies

Strategies:

- **ε-greedy**
  - With probability ε take a uniformly random action;
  - Otherwise take optimal action.
- **Boltzman**
  - Pick action proportionally to transformed Qvalues

$$P(a) = softmax\left(\frac{Q(a)}{std}\right)$$

- Policy based: add entropy
- Bandits or finite MDP: UCB-1

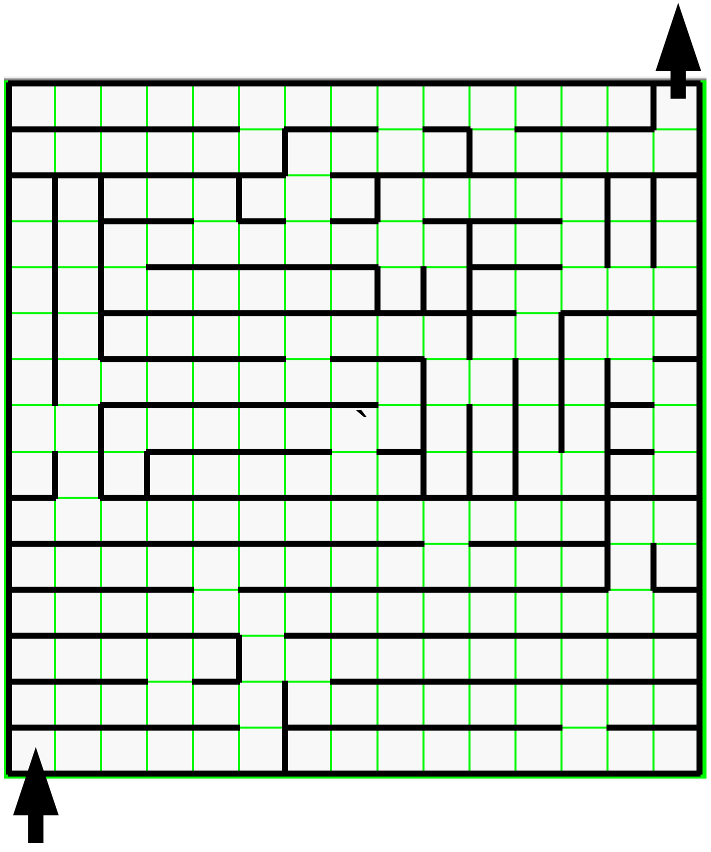
Voila! We've solved the reinforcement learning!

Or have we?

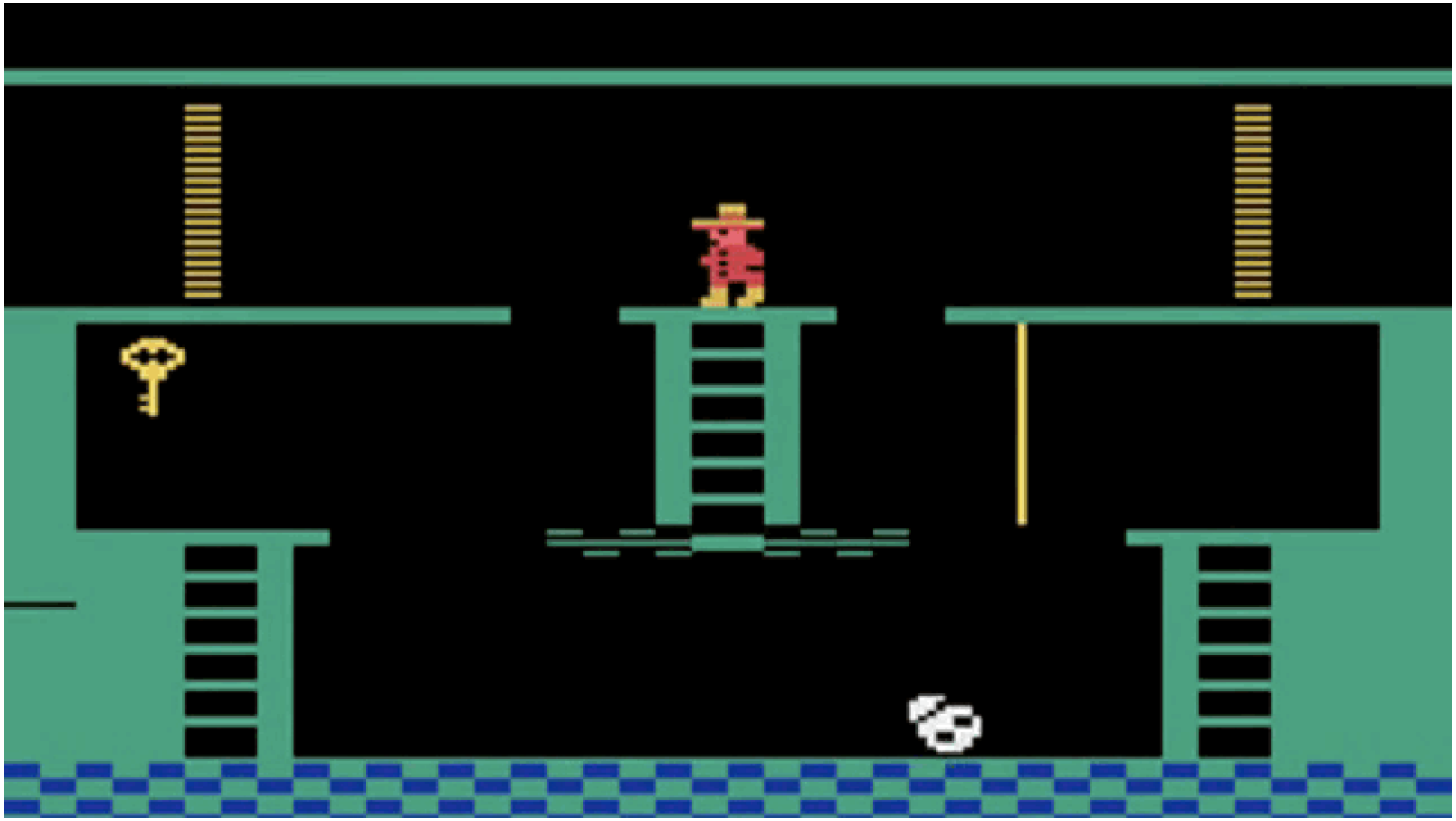
What happens if we apply it to real world problems?

How many random actions does it take to exit this maze?

(you only get reward at the end)



Less than it takes to discover that this game is solved by using the key



Less than it takes to learn how to

- Apply medical treatment
- Control robots
- Invent efficient VAE training

Except humans can learn these in less than a lifetime

Less than it takes to learn how to

- Apply medical treatment
- Control robots
- Invent efficient VAE training

We humans explore not with e-greedy policy!

BTW how humans explore?

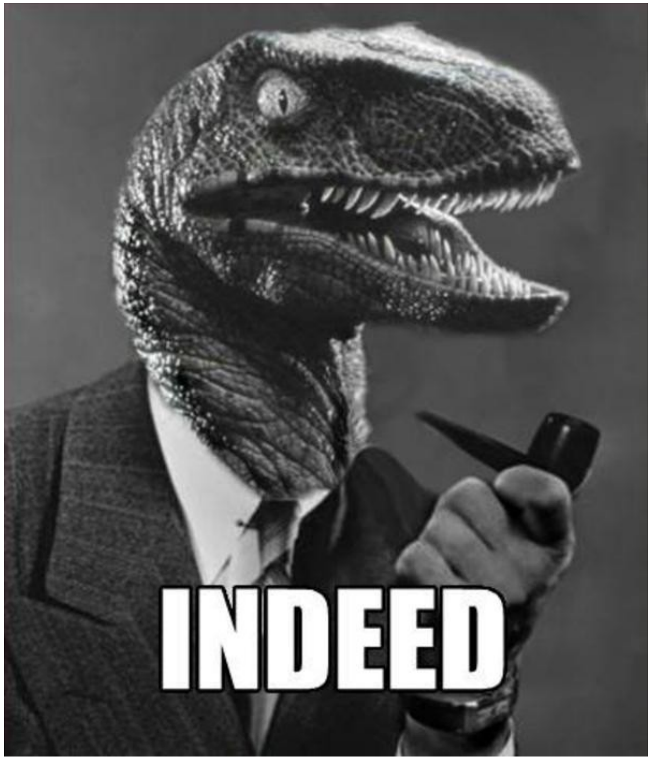
Whether some new particles violate physics

Vs

Whether you still can't fly by pulling your hair up

I got it!

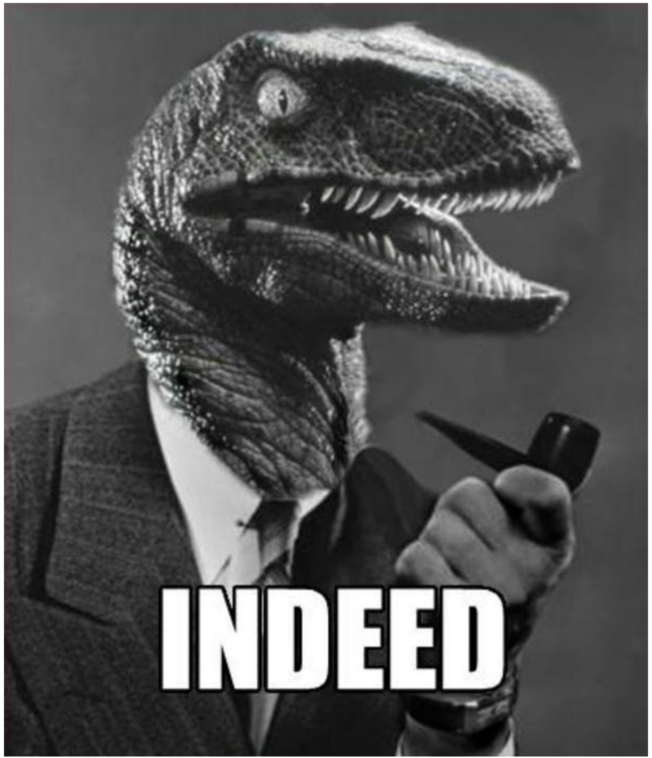
It's all about funding!



I got it!

It's all about funding!

(Just kidding)



Reward augmentation

Let's “pay” agent for exploration!

$$\tilde{r}(z,a,s')=r(s,a,s')+r_{exploration}(s,a,s')$$

Reward augmentation

Let's “pay” agent for exploration!

$$\tilde{r}(z,a,s')=r(s,a,s')+r_{exploration}(s,a,s')$$

Extrinsic reward **Intrinsic reward**

In Real Life

Extrinsic reward

– Not dying is good

Curiosity

– Discovering new things feels good

Social stuff

– Helping others feels good
– Being praised feels good

...

In Real Life

- Extrinsic reward
  - Not dying is good

Disclaimer: I am NOT an expert neurologist/psychologist :)

- Curiosity
  - Discovering new things feels good
- Social stuff
  - Helping others feels good
  - Being praised feels good

...

Reward augmentation

Let's “pay” agent for exploration!

$$\tilde{r}(z,a,s')=r(s,a,s')+r_{exploration}(s,a,s')$$

Trivia: Any suggestions on intrinsic r for atari/doom?

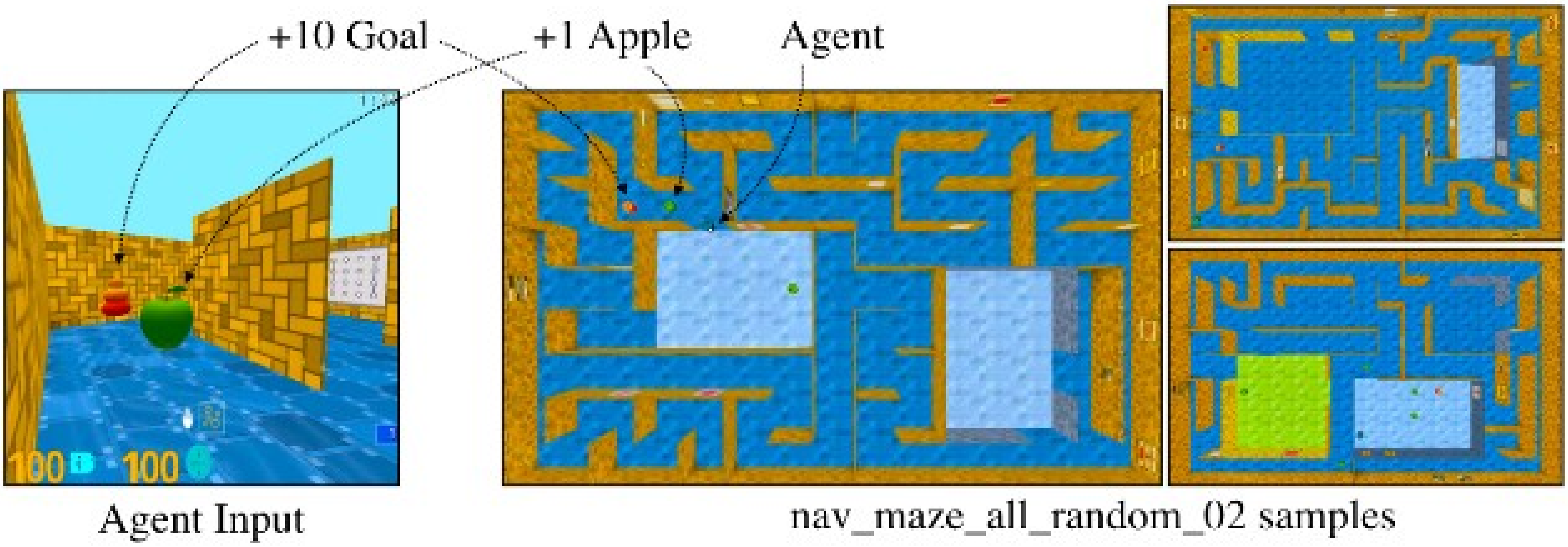
UNREAL main idea

- Auxilary objectives:
  - *Pixel control:* maximize pixel change in NxN grid over image
  - *Feature control:* maximize activation of some neuron deep inside neural network
  - *Reward prediction:* predict future rewards given history

article: arxiv.org/abs/1611.05397

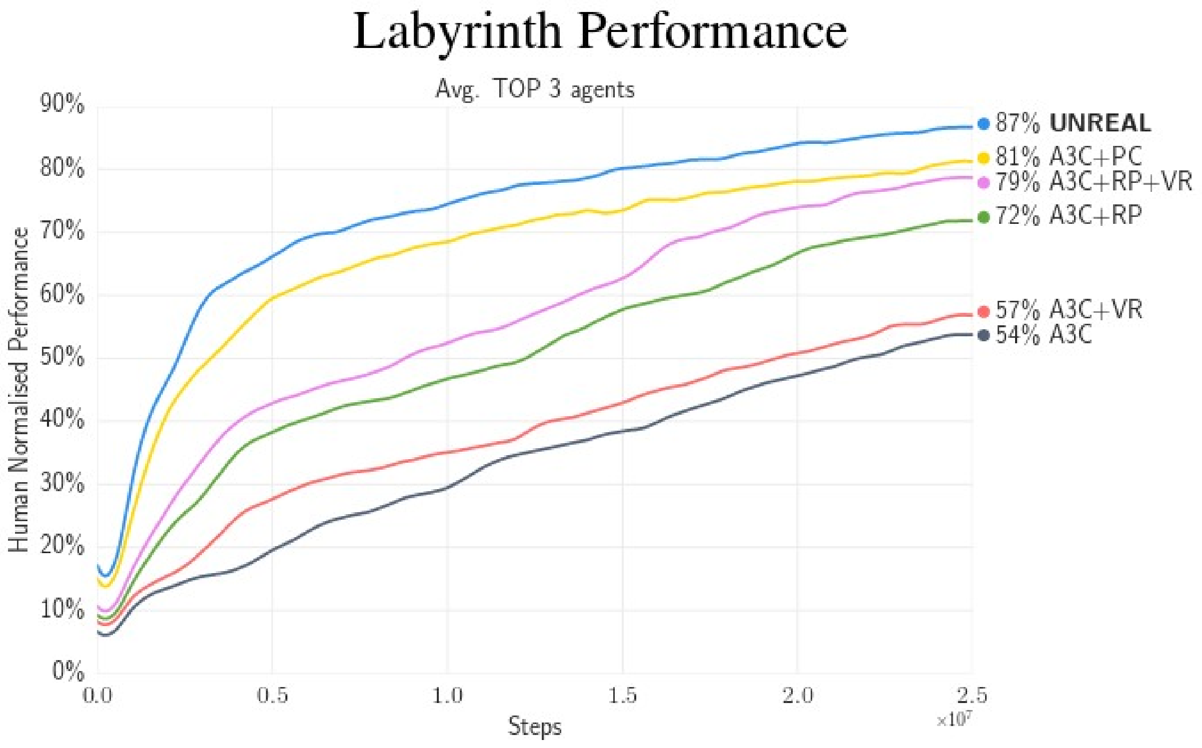
blog post: bit.ly/2g9Yv2A

Environment: Labyrinth

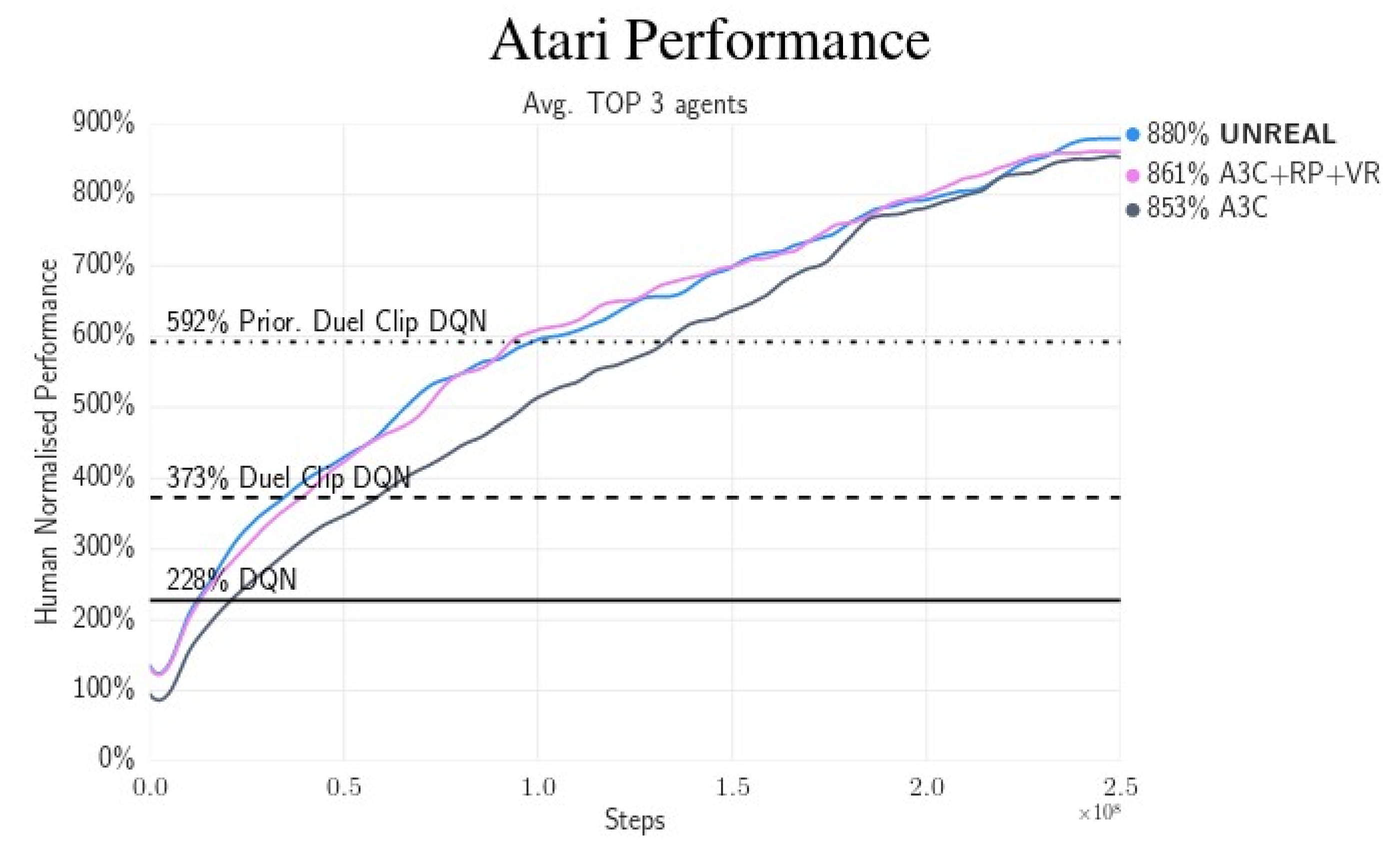


- Maze with rewards
- Partially observable
  - Used a2c + LSTM + experience replay

Results: labyrinth



Results: Atari



Count-based exploration

UCB-1 for bandits

Idea:

Prioritize actions with uncertain outcomes!

Less times visited = more uncertain.

Math: add upper confidence bond to reward.

Count-based exploration

UCB-1 for bandits

Take actions in in proportion to $\tilde{v}_a$

$$\tilde{v}_a = v_a + \sqrt{\frac{2 \log N}{n_a}}$$

Upper bound
For bernoilli r

- N number of time-steps so far
- n_a times action **a** is taken

Count-based exploration

UCB-1 for bandits

Take actions in in proportion to $\tilde{v}_a$

$$\tilde{v}_a = v_a + \sqrt{\frac{2 \log N}{n_a}}$$

– N number of time-steps so far

– n_a times action **a** is taken

Count-based exploration

UCB generalized for multiple states

$$\tilde{Q}(s,a) = Q(s,a) + \alpha \cdot \sqrt{\frac{2 \log N_s}{n_{s,a}}}$$

where

- N_s visits to state **s**
- $n_{s,a}$ times action **a** is taken from state **s**

Count-based models

TL;DR article

- Use approximate density model, p(s)
- Encourage visiting rare states

Article: arxiv.org/abs/1606.01868

Vime motivation

Curiosity

Taking actions that increase your knowledge about the world (a.k.a. the environment)

Knowledge about the world

Whatever allows you to predict how world works depending on your behavious

Vime main idea

Add curiosity to the reward

$$\tilde{r}(z, a, s') = r(s, a, s') + \beta r_{vime}(z, a, s')$$

Curiosity definition

$$r_{vime}(z, a, s') = I(\theta; s' | z, a)$$

Vime main idea

Environment model

$$P\left(s'|s,a,\theta\right)$$

Session

$$z_t = \langle s_{0,} a_{0,} s_{1,} a_{1,} \ldots, s_t \rangle$$

Surrogate reward

$$\tilde{r}\left(z,a,s'\right) = r\left(s,a,s'\right) + \beta r_{vime}\left(z,a,s'\right) = r\left(s,a,s'\right) + \beta I\left(\theta; s'|z,a\right)$$

curiosity

$$I\left(\theta; s'|z,a\right) = H\left(\theta|z,a\right) - H\left(\theta|z,a,s'\right) = E_{s_{t+1} \sim P\left(s_{t+1}|s,a\right)} KL\left[P\left(\theta|z,a,s'\right) \| P\left(\theta|z\right)\right]$$

need proof for that last line?

Naive objective

$$E_{s_{t+1}\sim P(s_{t+1}|s,a)} KL[P(\theta|z,a,s')\|P(\theta|z)] = \int\limits_{s'} P(s'|s,a)\cdot\int\limits_{\theta} P(\theta|z,a,s')\cdot\log\frac{P(\theta|z,a,s')}{P(\theta|z)}\,d\theta\,ds'$$

where

$$P(\theta|z)=\frac{P(z|\theta)\cdot P(\theta)}{P(z)}=\frac{\prod\limits_{t} P(s_{t+1}|s_t,a_t,\theta)\cdot P(\theta)}{\int\limits_{\theta} P(z|\theta)\cdot P(\theta)\,d\theta}$$

Naive objective

$$E_{s_{t+1}\sim P(s_{t+1}|s,a)} KL\left[P(\theta|z,a,s') \| P(\theta|z)\right] = \int_{s'} P(s'|s,a) \cdot \int_{\theta} P(\theta|z,a,s') \cdot \log \frac{P(\theta|z,a,s')}{P(\theta|z)} d\theta\, ds'$$

Sample From MDP **Sample Somehow...**

Model **Prior**

$$P(\theta|z) = \frac{P(z|\theta)\cdot P(\theta)}{P(z)} = \frac{\prod_t P(s_{t+1}|s_t, a_t, \theta)\cdot P(\theta)}{\int_\theta P(z|\theta)\cdot P(\theta)\, d\theta}$$

dunno

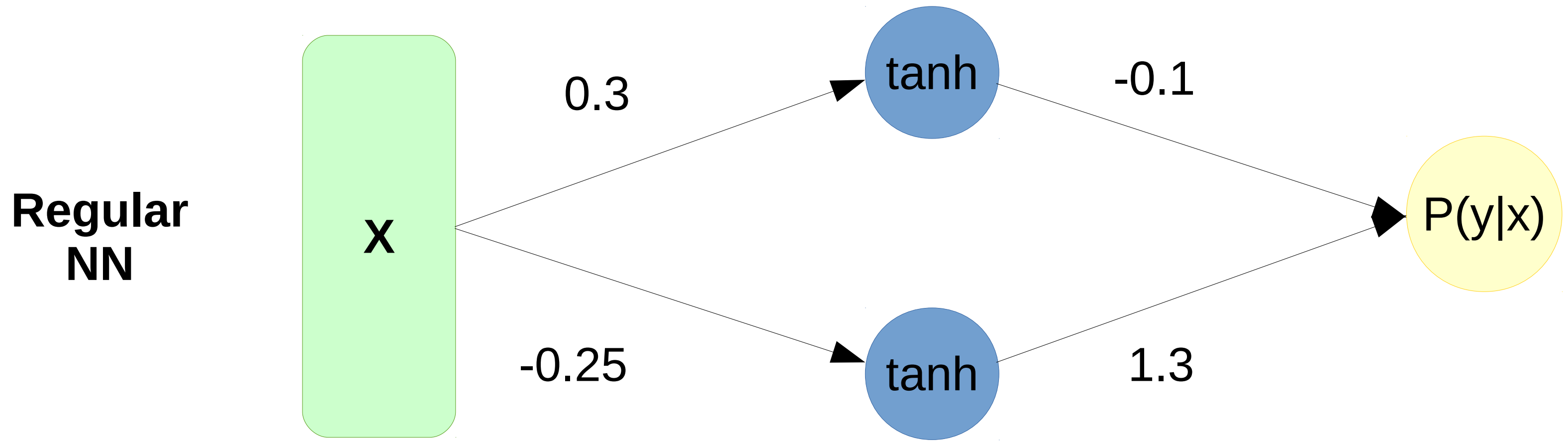
Better avoid computing P(theta|z) directly!


BAYES
TO SANTA ~~CLAUS~~

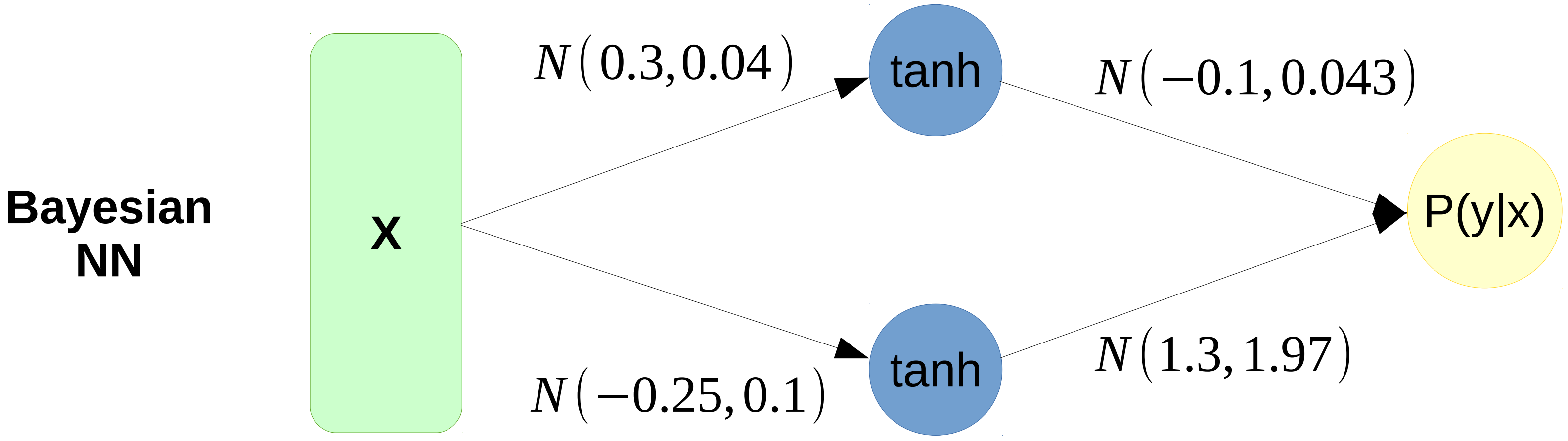
We want a model that

- predicts $P(s'|z,a,theta)$
- allows to estimate $P(theta|D)$
- we can sample from it

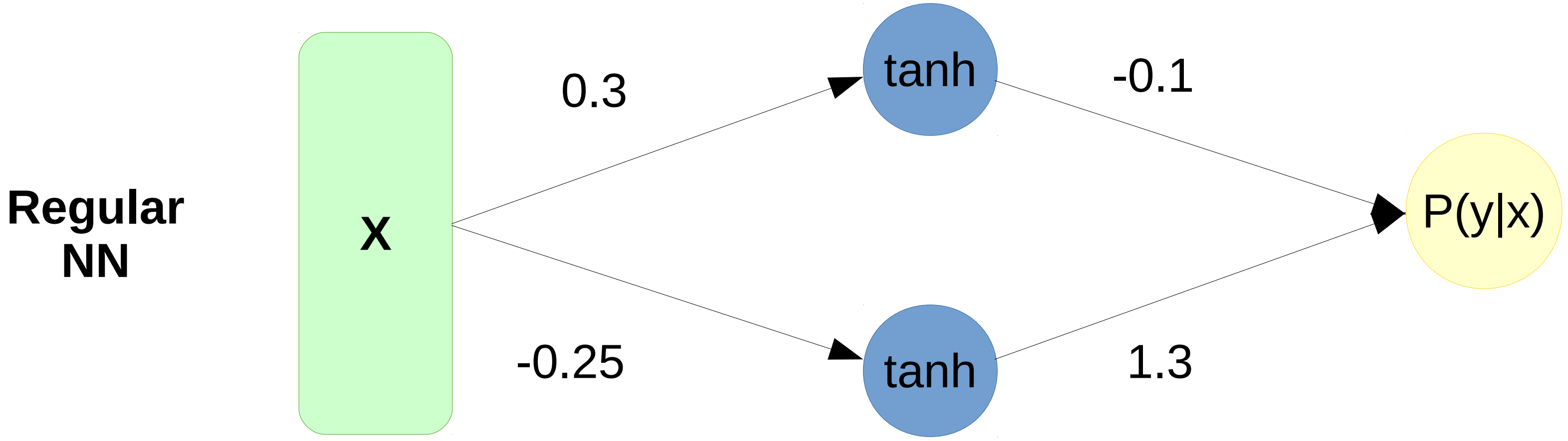
BNNs



BNNs

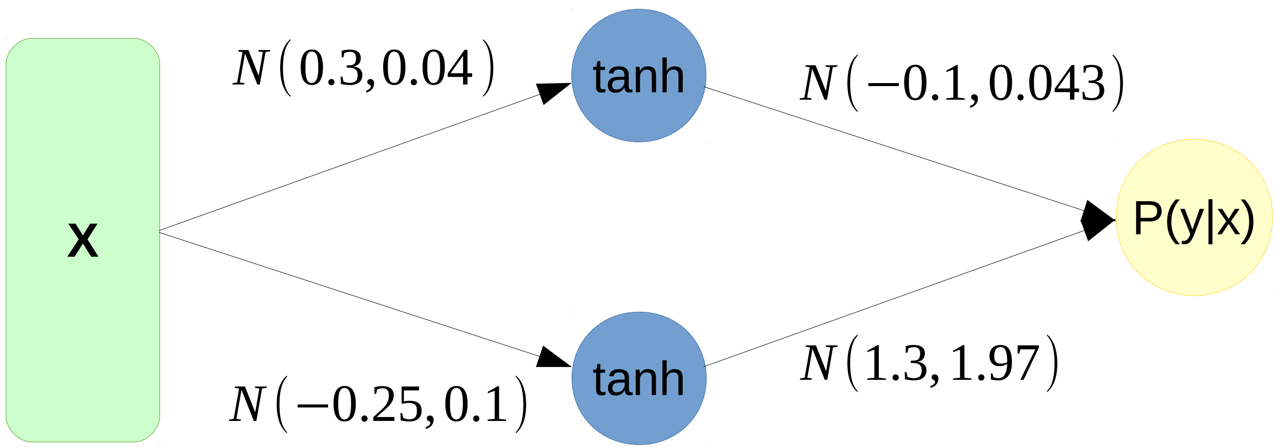


$N(0.3, 0.04)$
$N(-0.1, 0.043)$
Bayesian NN
X
tanh
P(y|x)
tanh
$N(-0.25, 0.1)$
$N(1.3, 1.97)$



0.3
-0.1
Regular NN
X
tanh
P(y|x)
tanh
-0.25
1.3

BNNs



Idea:

- No explicit weights
  - Maintain parametric distribution on them instead!
    - Practical: fully-factorized normal or similar

$$q(\theta|\varphi:[\mu,\sigma]) = \prod_i N(\theta_i|\mu_i,\sigma_i)$$

$$P(s'|s,a) = E_{\theta\sim q(\theta|\varphi)} P(s'|s,a,\theta)$$

BNNs

Idea:

- No explicit weights
  - Maintain parametric distribution on them instead!
    - Practical: fully-factorized normal or similar

$$q(\theta|\varphi:[\mu,\sigma])=\prod_i N(\theta_i|\mu_i,\sigma_i)$$

$$P(s'|s,a)=E_{\theta\sim q(\theta|\varphi)}P(s'|s,a,\theta)$$

- Learn parameters of that distribution (reparameterization trick)
  - Less variance: local reparameterization trick.

$$\mathring{\varphi}=argmax_{\varphi}E_{\theta\sim q(\theta|\varphi)}P(s'|s,a,\theta)$$

wanna explicit formulae?

Lower bound

$$\varphi_t = \underset{\varphi}{argmax}\left(-KL\left(q\left(\theta|\varphi\right)\|p\left(\theta|z_t\right)\right)\right)$$

$$\underset{\varphi}{argmax}\left(\left[E_{\theta\sim q(\theta|\varphi)}\log p\left(z_t|\theta\right)\right]-KL\left(q\left(\theta|\varphi\right)\|p\left(\theta\right)\right)\right)$$

Lower bound

$$-KL(q(\theta|\varphi)\|p(\theta|z))=-\int_{\theta} q(\theta|\varphi)\cdot\log\frac{q(\theta|\varphi)}{p(\theta|z)}$$

$$-\int_{\theta} q(\theta|\varphi)\cdot\log\frac{q(\theta|\varphi)}{[\frac{p(z|\theta)\cdot p(\theta)}{p(z)}]}=-\int_{\theta} q(\theta|\varphi)\cdot\log\frac{q(\theta|\varphi)\cdot p(z)}{p(z|\theta)\cdot p(\theta)}$$

$$-\int_{\theta} q(\theta|\varphi)\cdot[\log\frac{q(\theta|\varphi)}{p(\theta)}-\log p(z|\theta)+\log p(z)]$$

Trivia: what can you say about each of the summands?

Lower bound

$$-KL(q(\theta|\varphi)\|p(\theta|z)) = -\int_\theta q(\theta|\varphi)\cdot\log\frac{q(\theta|\varphi)}{p(\theta|z)}$$

$$-\int_\theta q(\theta|\varphi)\cdot\log\frac{q(\theta|\varphi)}{\left[\frac{p(z|\theta)\cdot p(\theta)}{p(z)}\right]} = -\int_\theta q(\theta|\varphi)\cdot\log\frac{q(\theta|\varphi)\cdot p(z)}{p(z|\theta)\cdot p(\theta)}$$

$$-\int_\theta q(\theta|\varphi)\cdot\left[\log\frac{q(\theta|\varphi)}{p(\theta)} - \log p(z|\theta) + \log p(z)\right]$$

$$[E_{\theta\sim q(\theta|\varphi)}\log p(z|\theta)] - KL(q(\theta|\varphi)\|p(\theta)) + \log p(z)$$

loglikelihood **-distance to prior** **+const**

Lower bound

$$\varphi_t = \underset{\varphi}{argmax}\left(-KL\left(q\left(\theta|\varphi\right) \| p\left(\theta|z_t\right)\right)\right)$$

$$\underset{\varphi}{argmax}\left(\left[E_{\theta \sim q\left(\theta|\varphi\right)} \log p\left(z_t|\theta\right)\right] - KL\left(q\left(\theta|\varphi\right) \| p\left(\theta\right)\right)\right)$$

Can we perform gradient ascent directly?

Reparameterization trick

$$\varphi_t = \underset{\varphi}{argmax}\left(-KL\left(q(\theta|\varphi)\,\|\,p(\theta|z_t)\right)\right)$$

$$\underset{\varphi}{argmax}\left(\underline{\left[E_{\theta\sim q(\theta|\varphi)}\log p(z_t|\theta)\right]}-KL\left(q(\theta|\varphi)\,\|\,p(\theta)\right)\right)$$

Use reparameterization trick

Using BNN

$$E_{\theta\sim N(\theta|\mu_\varphi,\sigma_\varphi)}\log p(z|\theta) = E_{\psi\sim N(0,1)}\log p(z|(\mu_\varphi+\sigma_\varphi\cdot\psi))$$

Better: local reparameterization trick (google it)

Vime objective

$$E_{s_{t+1} \sim P(s_{t+1}|s,a)} KL\left[P(\theta|z,a,s') \| P(\theta|z)\right] = \int_{s'} P(s'|s,a) \cdot \int_{\theta} P(\theta|z,a,s') \cdot \log \frac{P(\theta|z,a,s')}{P(\theta|z)} d\theta\, ds'$$

$$KL\left[P(\theta|z,a,s') \| P(\theta|z)\right] \approx KL\left[q(\theta|z,a,s') \| q(\theta|z)\right] \equiv KL\left[q(\theta|\varphi') \| q(\theta|\varphi)\right]$$

$$E_{s_{t+1} \sim P(s_{t+1}|s,a)} KL\left[P(\theta|z,a,s') \| P(\theta|z)\right] \approx \int_{s'} \underbrace{P(s'|s,a)}_{\text{sample from env}} \cdot \int_{\theta} \underbrace{q(\theta|z,a,s')}_{\text{sample from BNN}} \cdot \log \frac{\overset{\text{BNN}}{q(\theta|z,a,s')}}{\underset{\text{BNN last tick}}{q(\theta|z)}} d\theta\, ds'$$

Algorithm

Forever:

1. Interact with environment, get <s,a,r,s'>
2. Compute curiosity reward

$$\tilde{r}(z,a,s')=r(s,a,s')+\beta KL\left[q(\theta|\varphi')\|q(\theta|\varphi)\right]$$

3. train_agent(s,a,$\tilde{r}$,s') //with any RL algorithm
4. train_BNN(s,a,s') //maximize lower bound

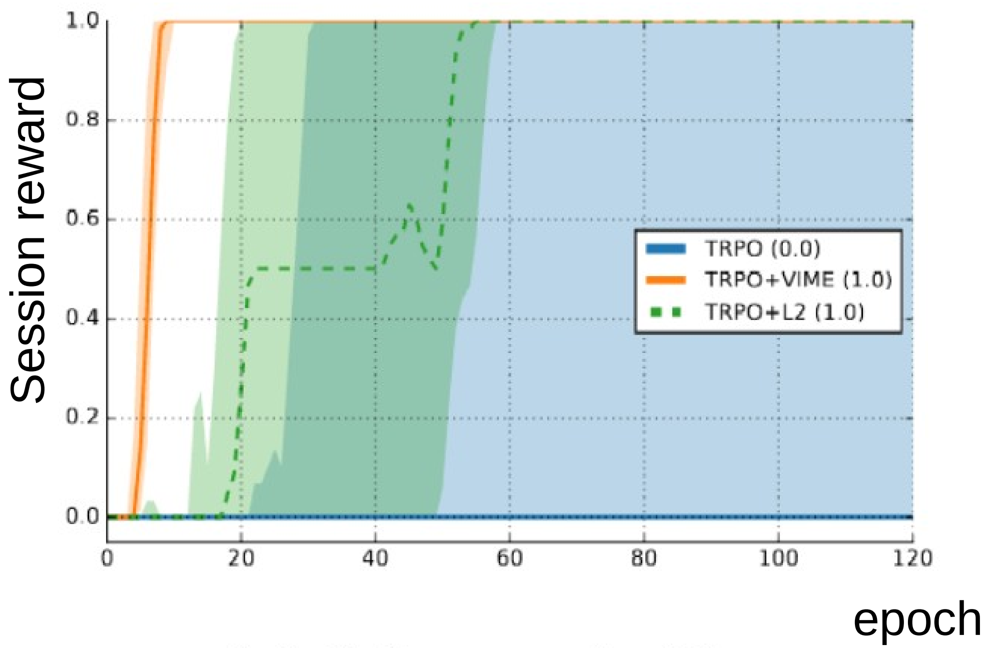
Dirty hacks

- **Use batches** of many <s,a,r,s'>
  - for CPU/GPU efficiency
  - greatly improves RL stability
- **Simple formula for KL**
  - Assuming fully-factorized normal distribution

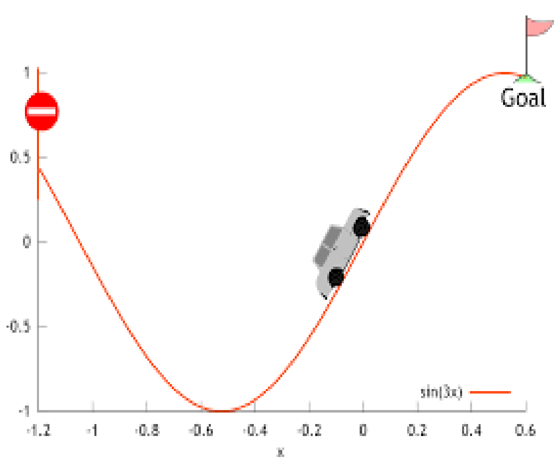
$$KL[q(\theta|\varphi')\|q(\theta|\varphi)]=\frac{1}{2}\sum_{i<|\theta|}[(\frac{\sigma_i'}{\sigma_i})^2+2\log\sigma_i-2\log\sigma_i'+\frac{(\mu_i'-\mu_i)^2}{\sigma_i^2}]$$

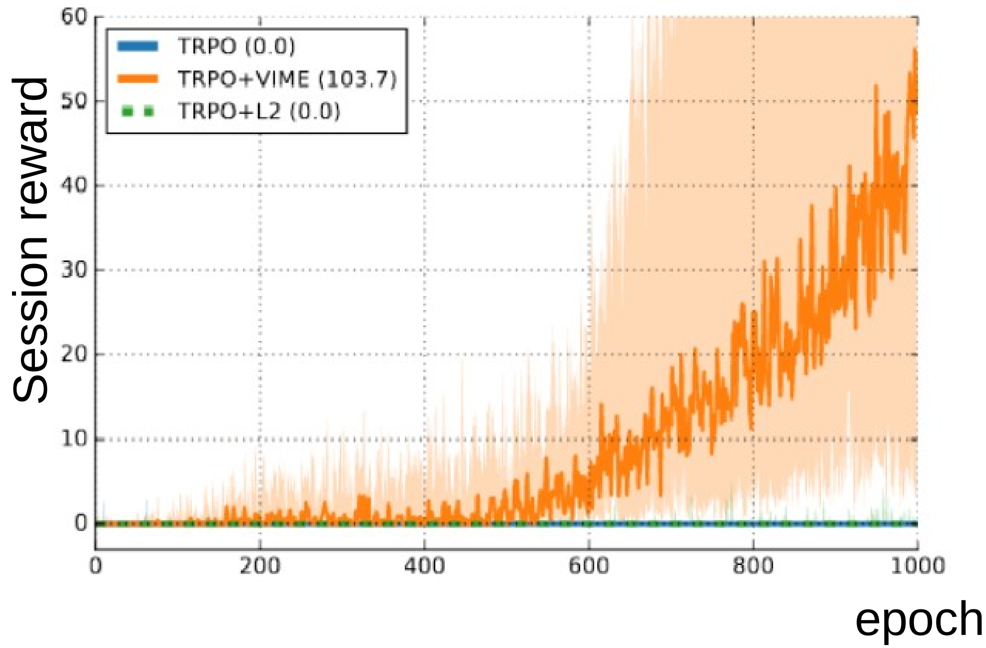
  - Even simpler: second order Taylor approximation
- **Divide KL by its running average over past iterations**

Results

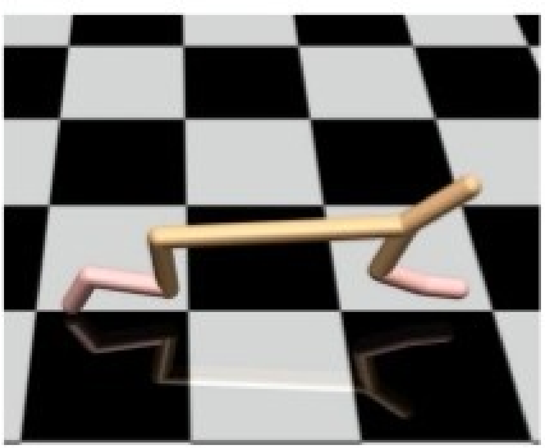


(a) MountainCar





(c) HalfCheetah



Results

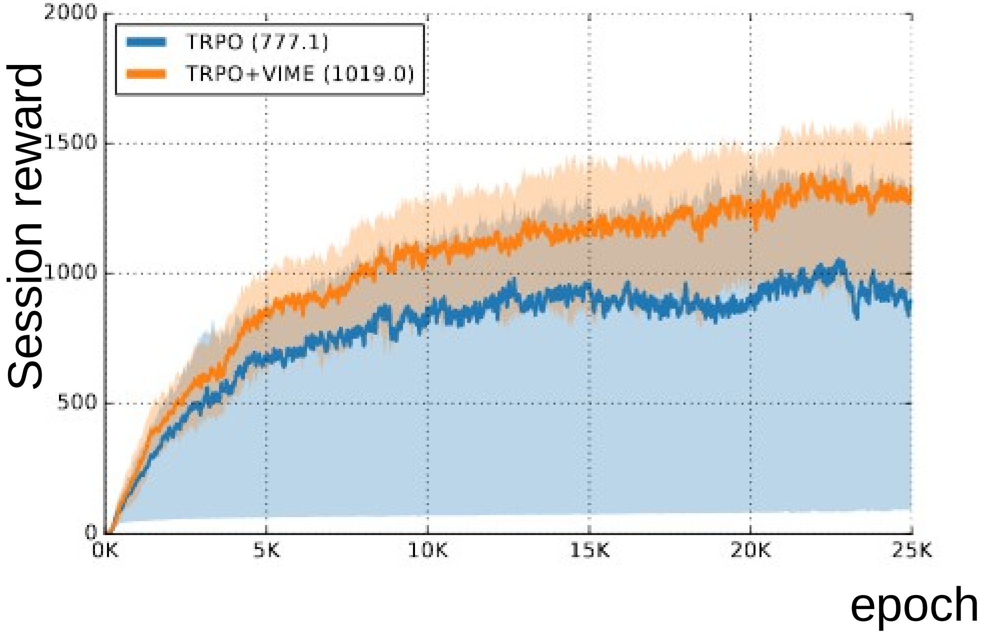
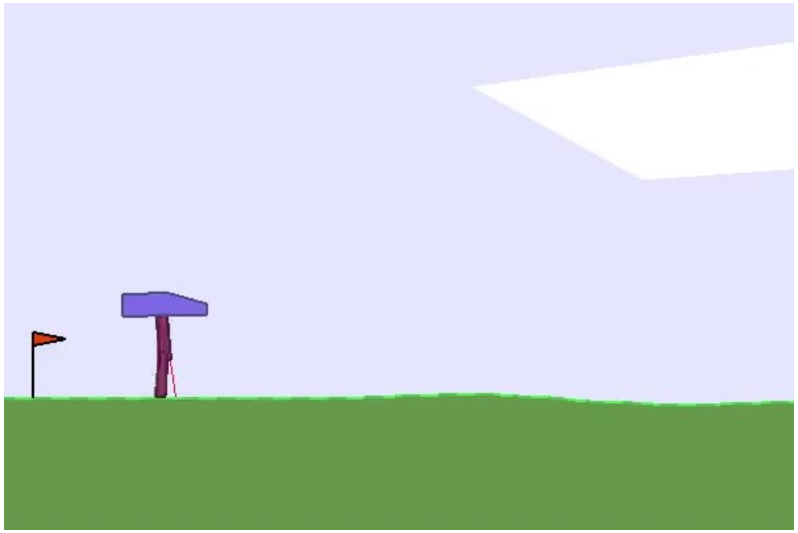
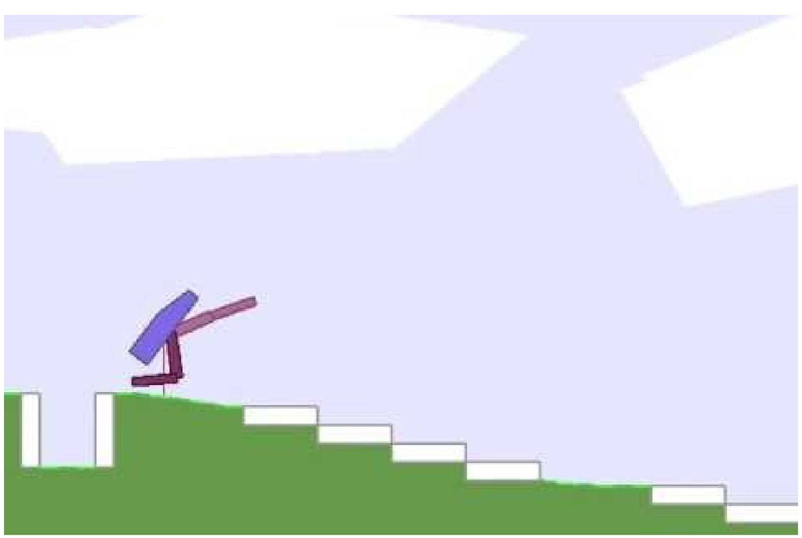


Figure 3: Performance of TRPO with and without VIME on the high-dimensional Walker2D locomotion task.

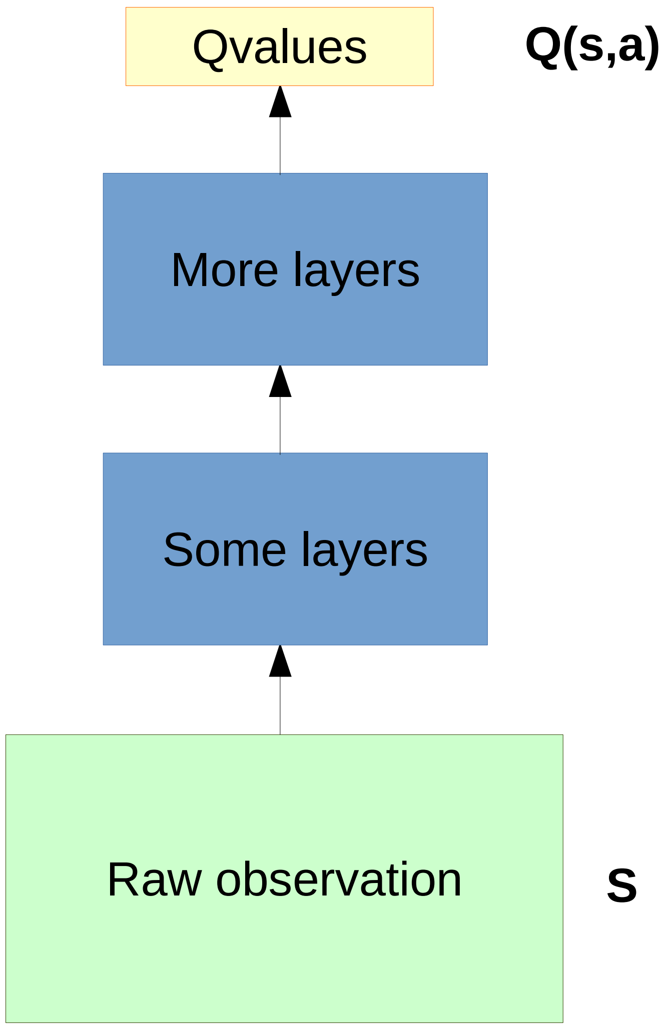




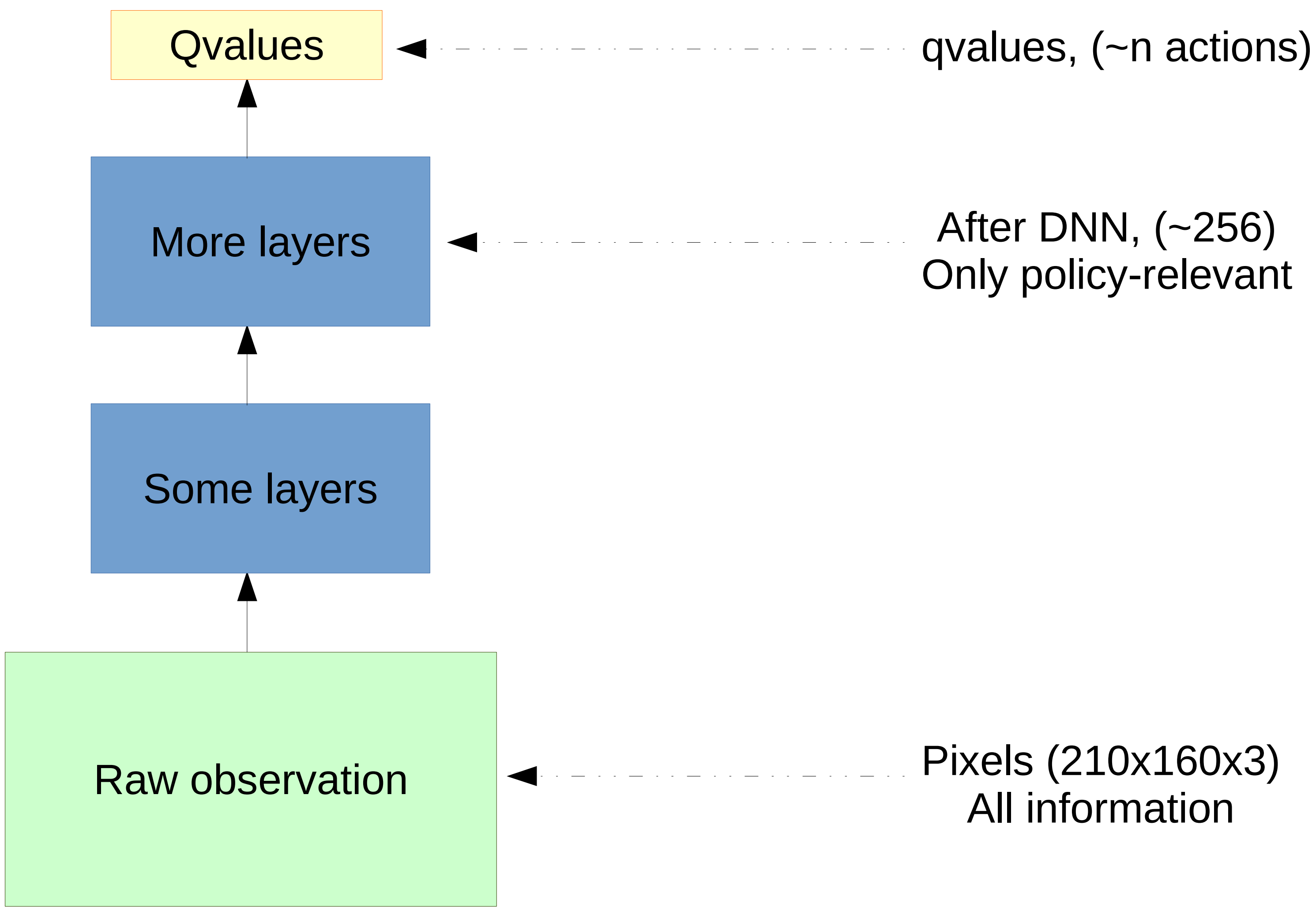
Pitfalls

- It's curious about irrelevant things
- Predicting (210x160x3) images is hard
- We don't observe full states (POMDP)

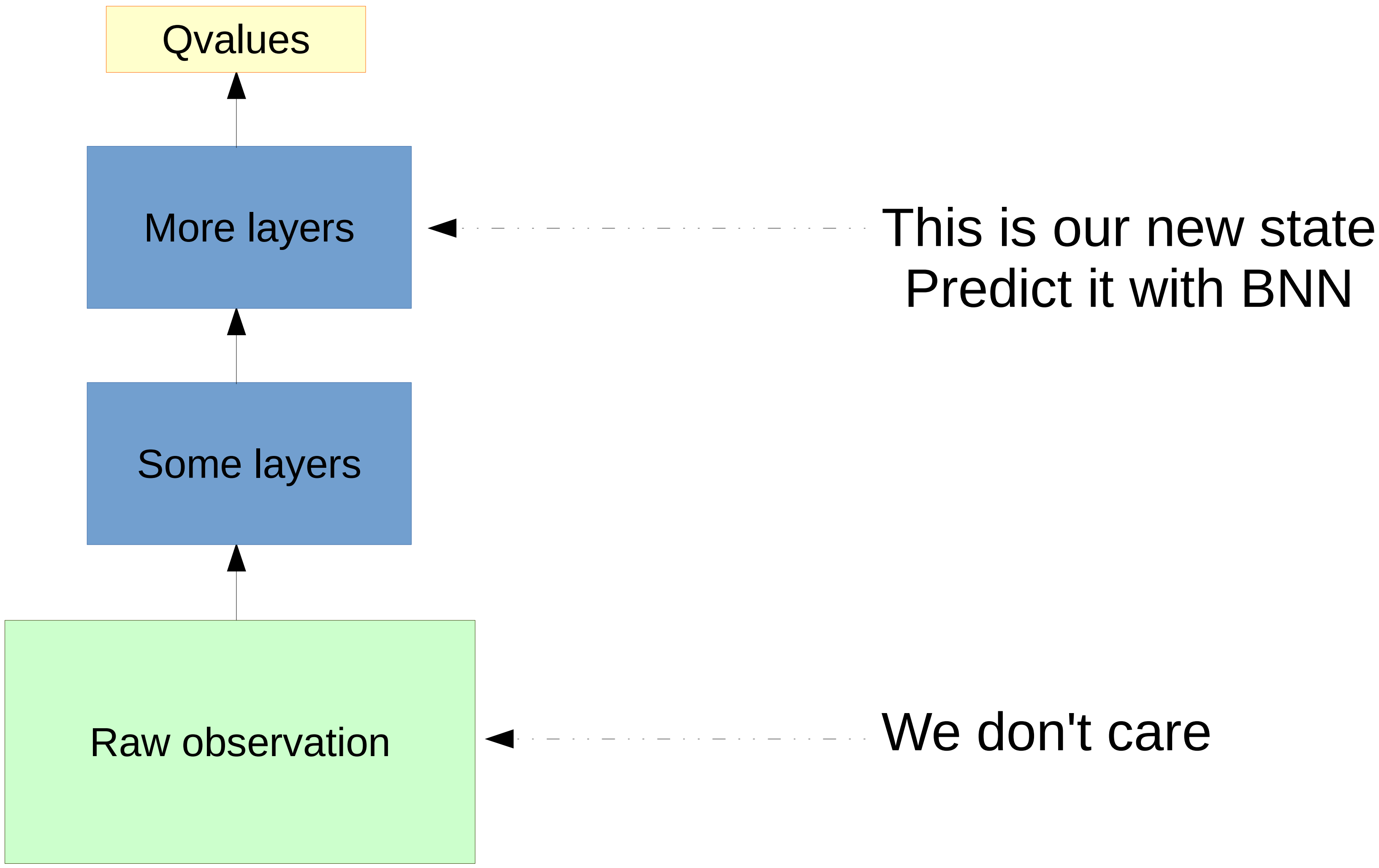
State = hidden NN activation



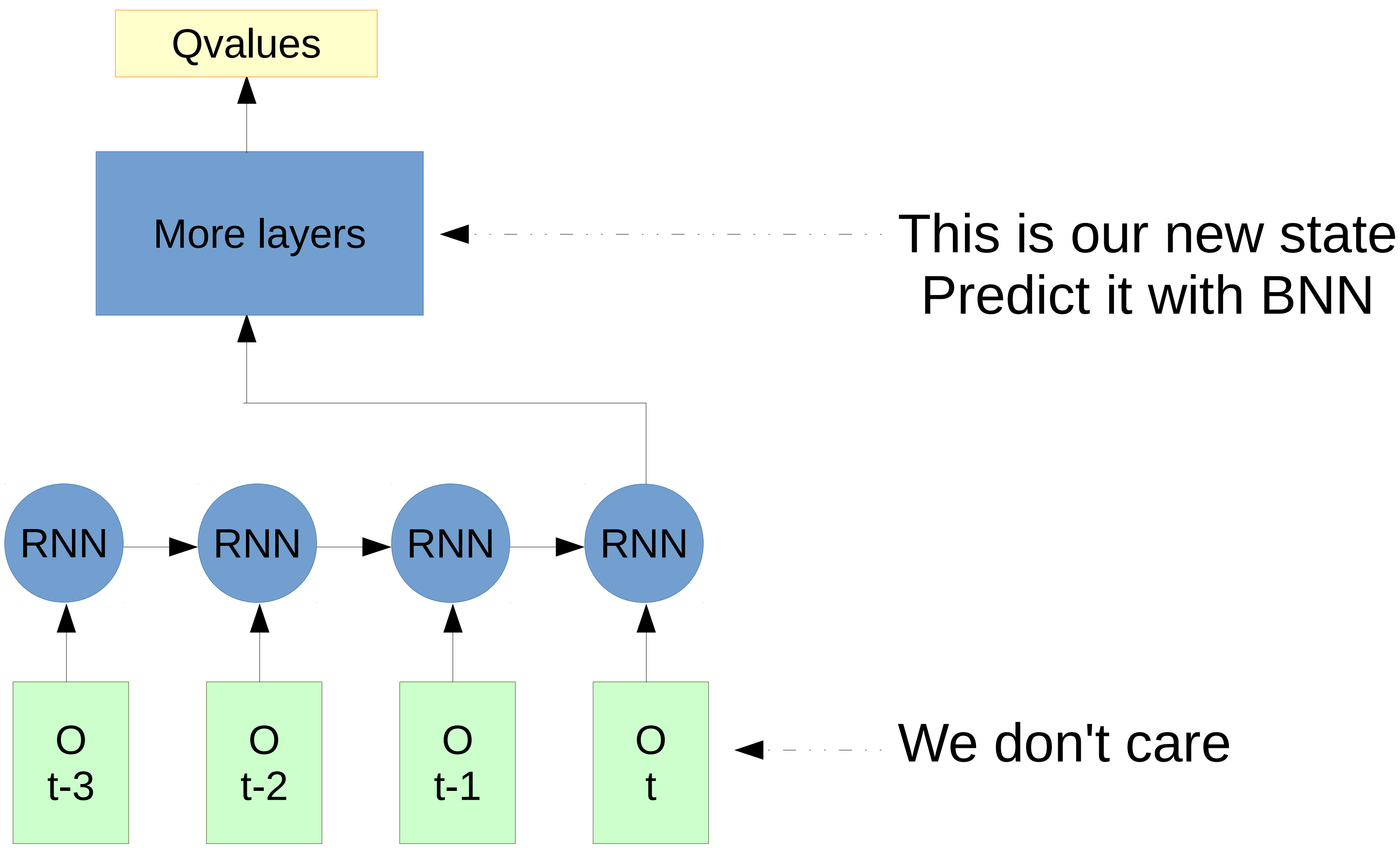
State = hidden NN activation



State = hidden NN activation



A case for POMDP



links

- https://arxiv.org/pdf/1605.09674v3.pdf
- http://mybinder.org/repo/justheuristic/vime
  - Will add pacman once it converges :P

Special thanks to

- **Arseniy Ashukha** – for reviewing (and covering my arse at HSE DL this very moment)
- **Maxim Kochurov** (ferrine@github) – for simple BNN @lasagne

Yandex
Data Factory

LAMBDA

British Hedgehog
Preservation Society