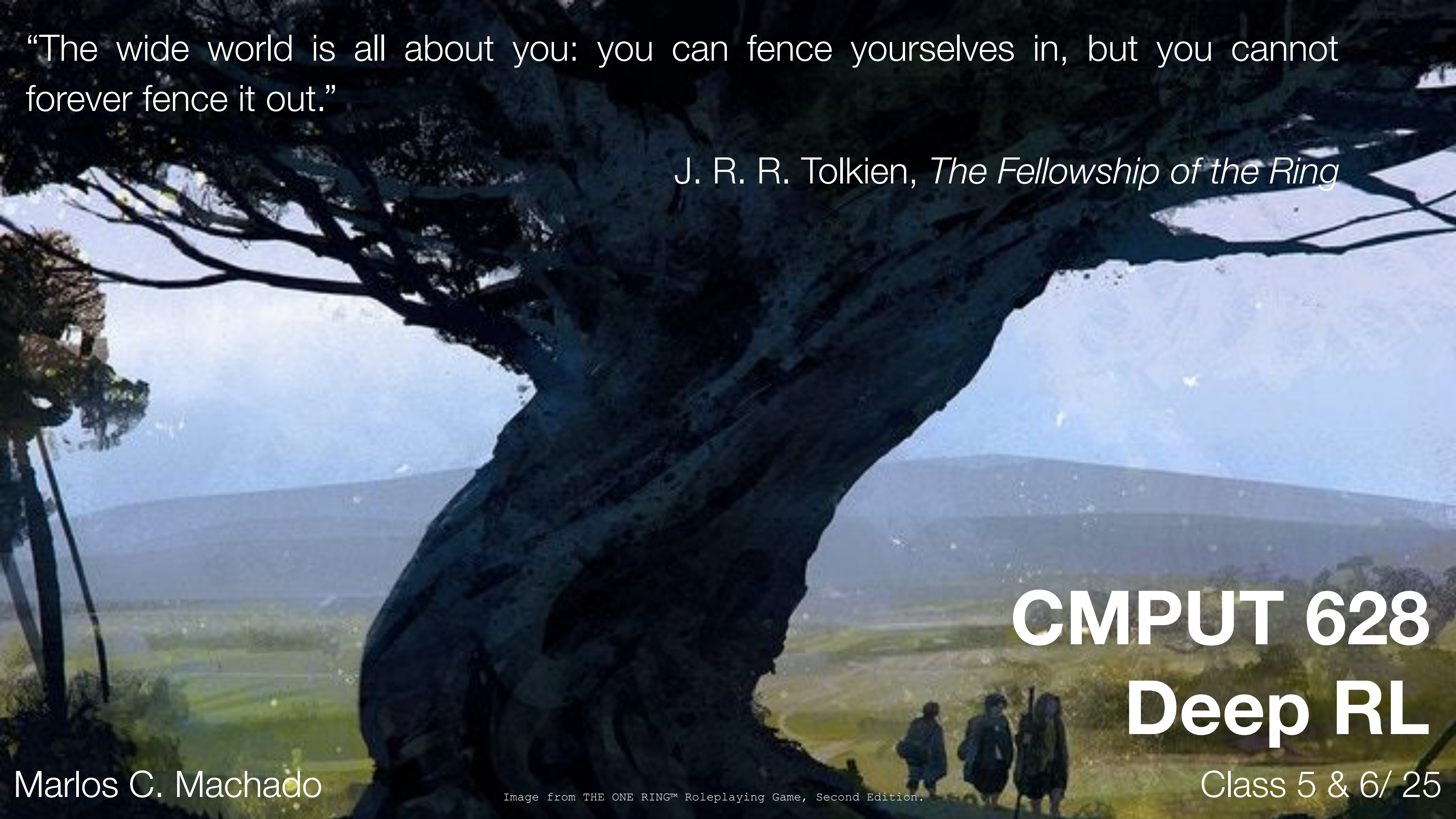


“The wide world is all about you: you can fence yourselves in, but you cannot forever fence it out.”

J. R. R. Tolkien, *The Fellowship of the Ring*

CMPUT 628 Deep RL

Marlos C. Machado

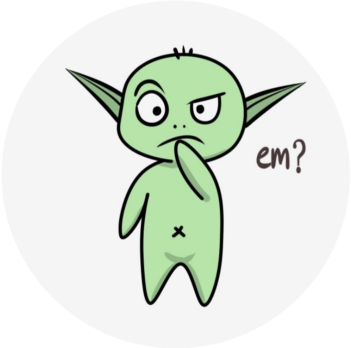
Image from THE ONE RING™ Roleplaying Game, Second Edition.

Class 5 & 6/ 25

Plan

The general structure of value-based model-free deep reinforcement learning methods; and DQN.

Please, interrupt me at any time!



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Reminders & Notes

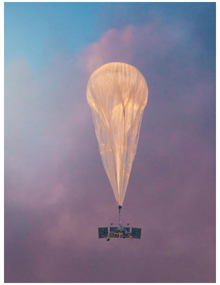
- You can’t leave anymore 😃
- Assignment 1 is due this Saturday, Jan. 24, 2026
- We will release assignment 2 next week, after the ICML deadline (Jan. 28, 2025) ¯_(ツ)_/¯
- I will release instructions about seminar and paper review during the reading week (Feb 16 – Feb 20)

Deep RL is about Function Approximation

- Why use function approximation?

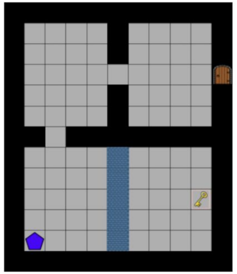
“Underparameterization”

# params < num states



“Overparameterization”

# params > num states



Deep RL is about Function Approximation

- Why use function approximation?
  - Scalability and generalization
- What is bigger? The number of parameters in the NN or the number of states?

$$\boldsymbol{\theta} \in \mathbb{R}^d \qquad q_\pi(s, a; \boldsymbol{\theta}) \approx q_\pi(s, a) \qquad d \ll |\mathcal{S}|$$

"Underparameterization"

# params < num states

– Optimality is impossible, the blanket is too small

– We should talk about max. sum of (discounted) rewards

You should try to identify which case you are in

"Overparameterization"

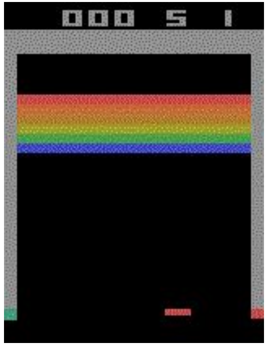
# params > num states

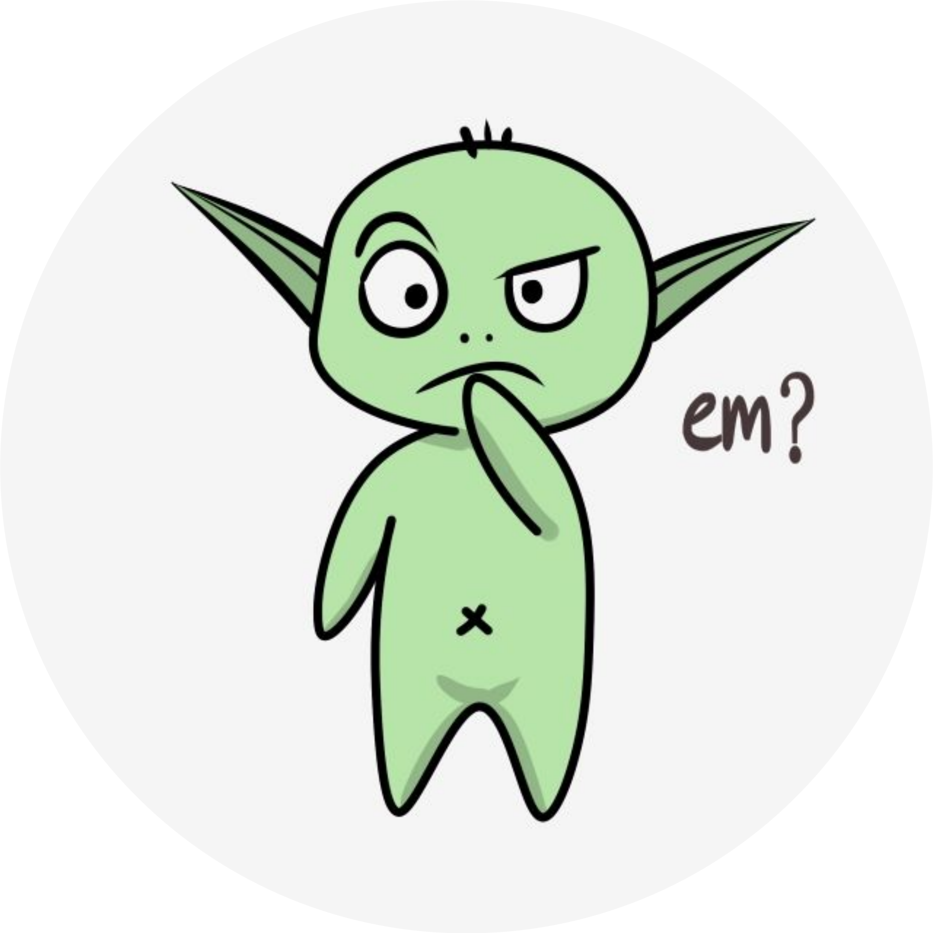
– It is about learning a mapping from high-dimensional obs. to the underlying (small) state space.

– Often formalized as a Block MDP

Observation, Agent State, Environment State

- Atari screens are drawn with 128 colours in a 160 x 210 resolution.
  - There are $128^{210x160}$ (10^{76220}) possible images!
  - The Atari 2600 console only has 128 bytes of RAM.
- The number of states in Breakout is estimated to be between 10^9 and 10^{11}
  - If we use 32 bits to represent a state, it could take up to 400 GB!
  - But for comparison, the number of states Chess and Go are estimated to have is 10^{46} and 10^{172}, respectively.
- The observation is not a state. There are two types of state: the environment state and the agent state.



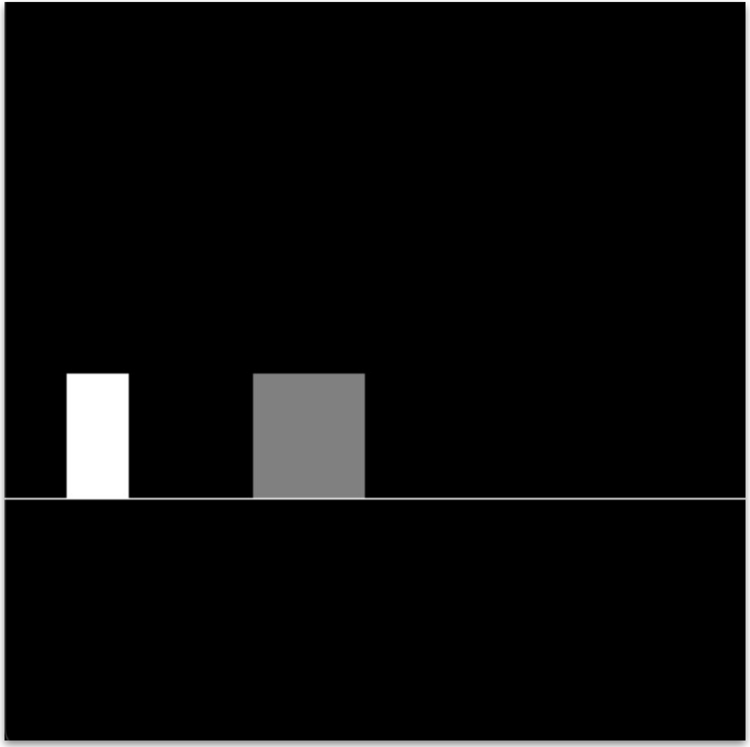


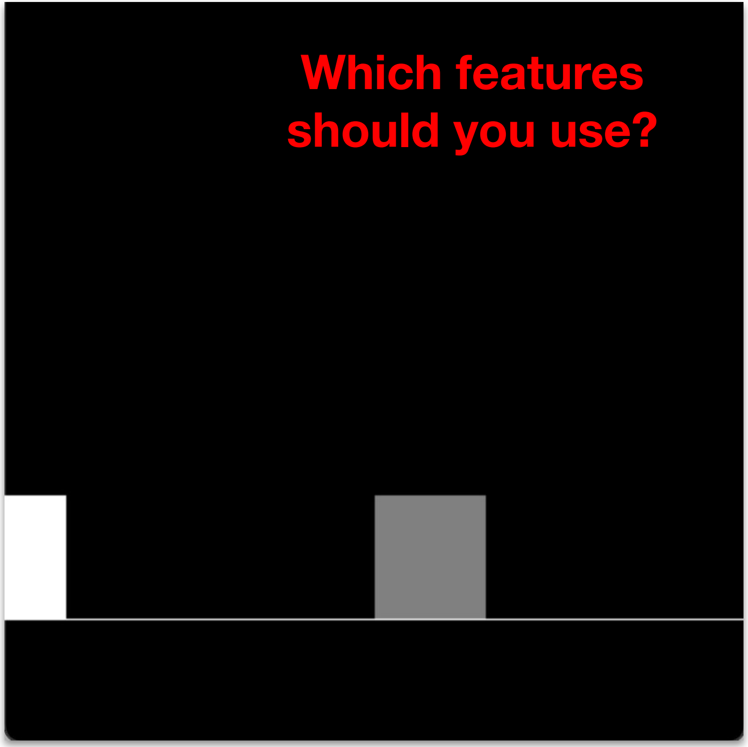
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Function Approximation

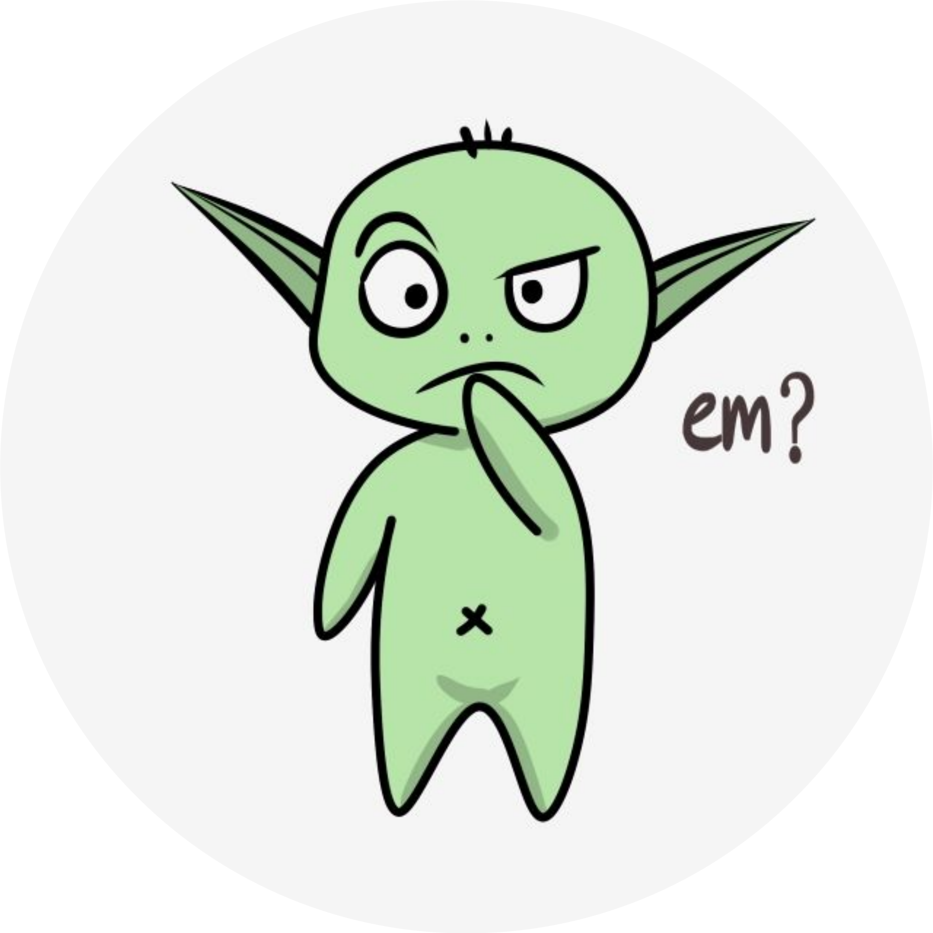
- In these next classes we'll be approximating value functions, later we will talk about approximating models or directly approximating policies.
- We are studying deep reinforcement learning because of its scalability and its ability (or promise) to learn representations.
- It is easy to take for granted the representation learning aspect, this is why your assignment 2 is partly about designing features :-)

Example: Jumping Task [Tachet des Combes et al., 2018]





Which features should you use?



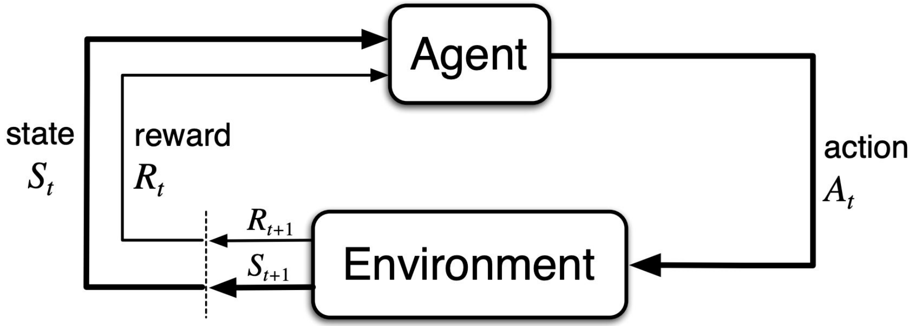
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

The Main Components of Value-based Model-free Methods

- At least for now, deep RL algorithms are more than RL algorithms with NNs
- In deep RL, the agent-environment interaction *is the same* as in traditional RL, but deep RL agents tend to be quite different from RL agents.
    - They tend to have many more components, they are more complicated

… but before that…

In traditional RL, how would we use NNs for funct. approx.?



$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha\Big[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(O_{t+1}, a'; \boldsymbol{\theta}_t) - Q(O_t, A_t; \boldsymbol{\theta}_t)\Big]\nabla_{\boldsymbol{\theta}_t} Q(O_t, A_t; \boldsymbol{\theta}_t)$$

Why don’t we use this?

Online Q-Learning with Neural Networks (OQLNN)

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha\Big[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(O_{t+1}, a'; \boldsymbol{\theta}_t) - Q(O_t, A_t; \boldsymbol{\theta}_t)\Big]\nabla_{\boldsymbol{\theta}_t} Q(O_t, A_t; \boldsymbol{\theta}_t)$$

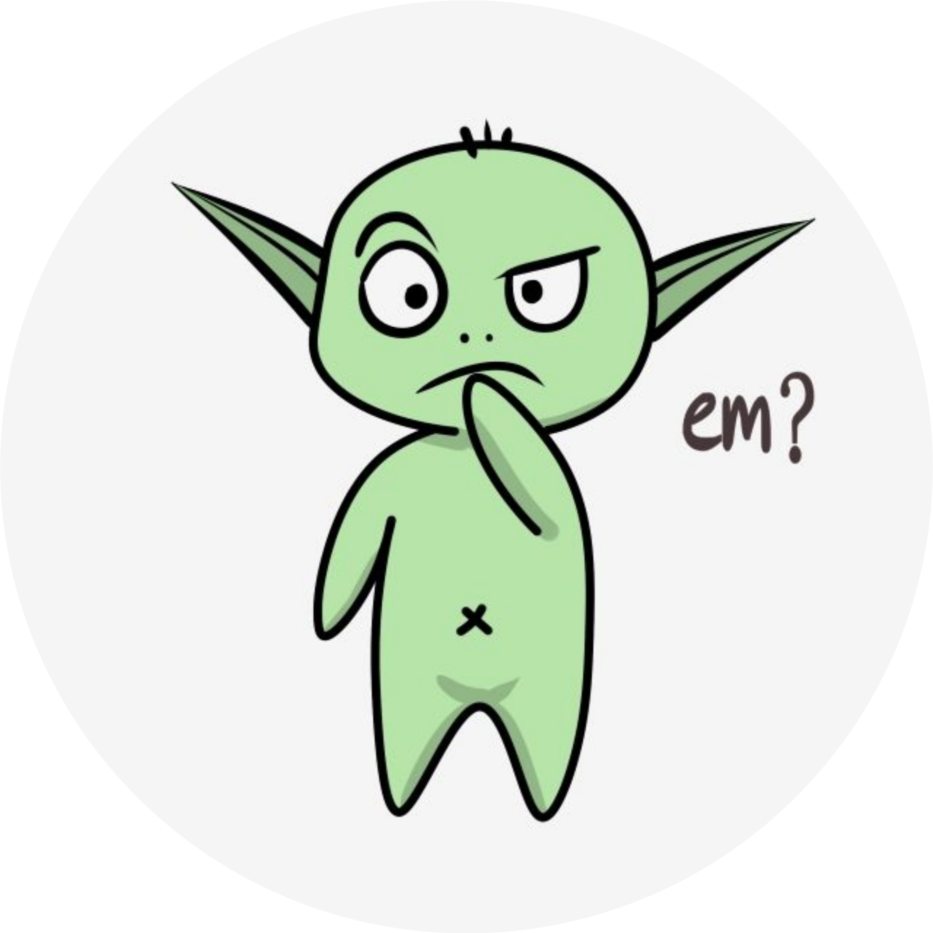
- It can be quite unstable
    - Consecutive samples end up being quite correlated
    - Generalization between the current and the next state can be quite dangerous for bootstrapping
- We don't *have to* process one sample at a time
    - We can't parallelize learning with one sample at a time
    - We might want to use a sample more than once

The Deadly Triad

- Instabilities arise when you combine:
  - Function approximation,
  - Bootstrapping, and
  - Off-policy learning.
- This happens even for prediction with linear function approximation.
- We are far from having theoretical results really justifying deep RL, so the vast majority of claims I'll make here will be based on empirical data.

For a manuscript with an actual empirical analysis about this issue in deep RL, see reference below.

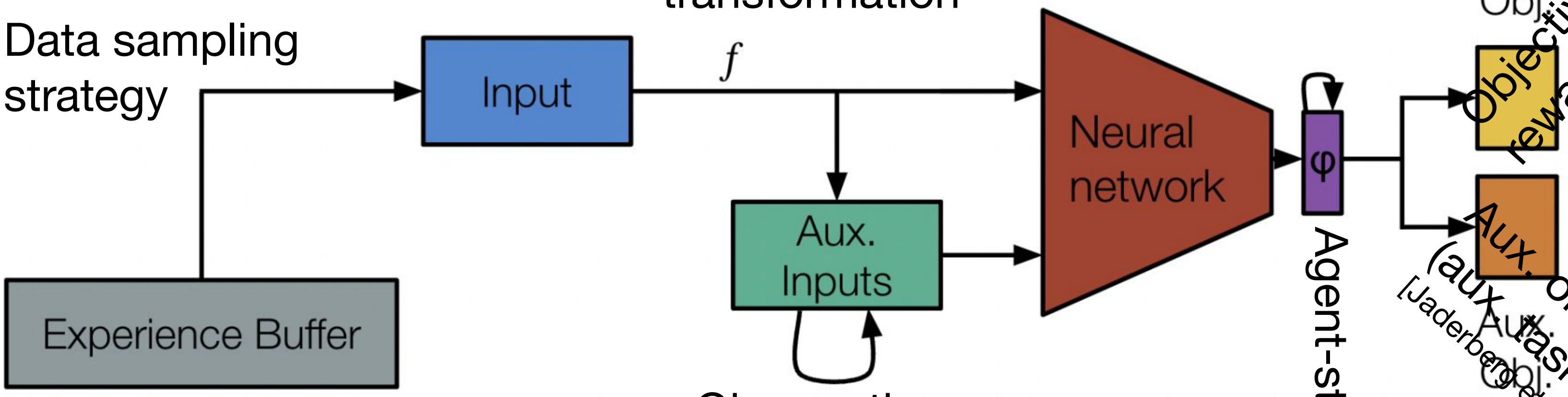
Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, Joseph Modayil: Deep Reinforcement Learning and the Deadly Triad. CoRR abs/1812.02648 (2018)



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Deep Reinforcement Learning

Structural Outline of Model-Free Value-based Deep RL Algs.



Data sampling strategy

Data transformation

Backbone of the neural network

Objective func. reward max.

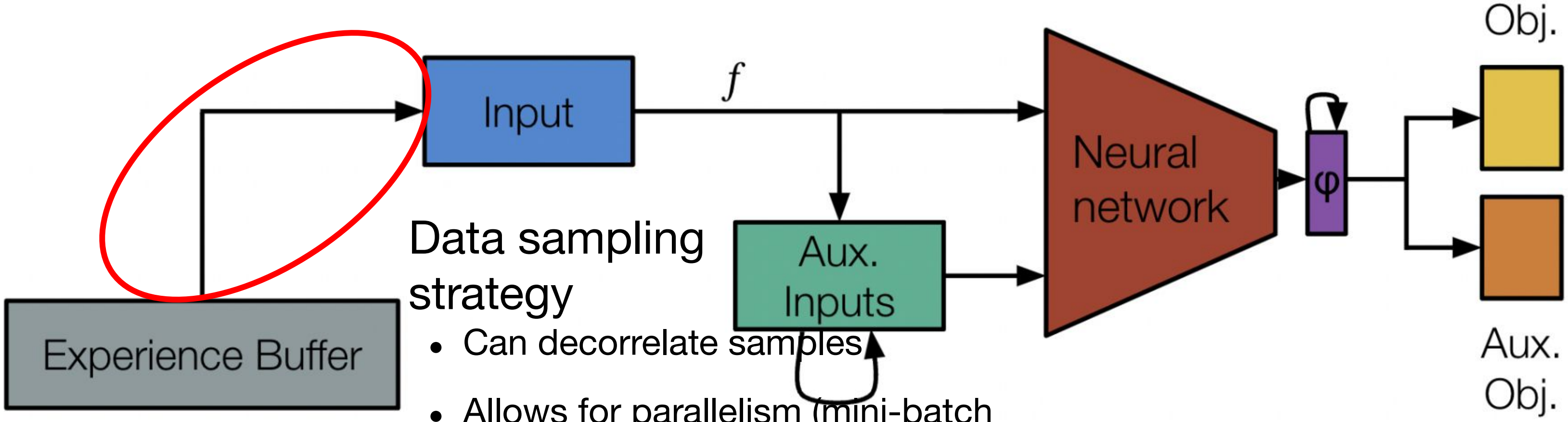
Agent-state

Aux. obj. func. (aux. tasks) [Jaderberg et al., 2017]

Fixed size buffer to store transitions (often FIFO) [Lin et al., 1991]

Observation augmentation [Tao et al., 2023]

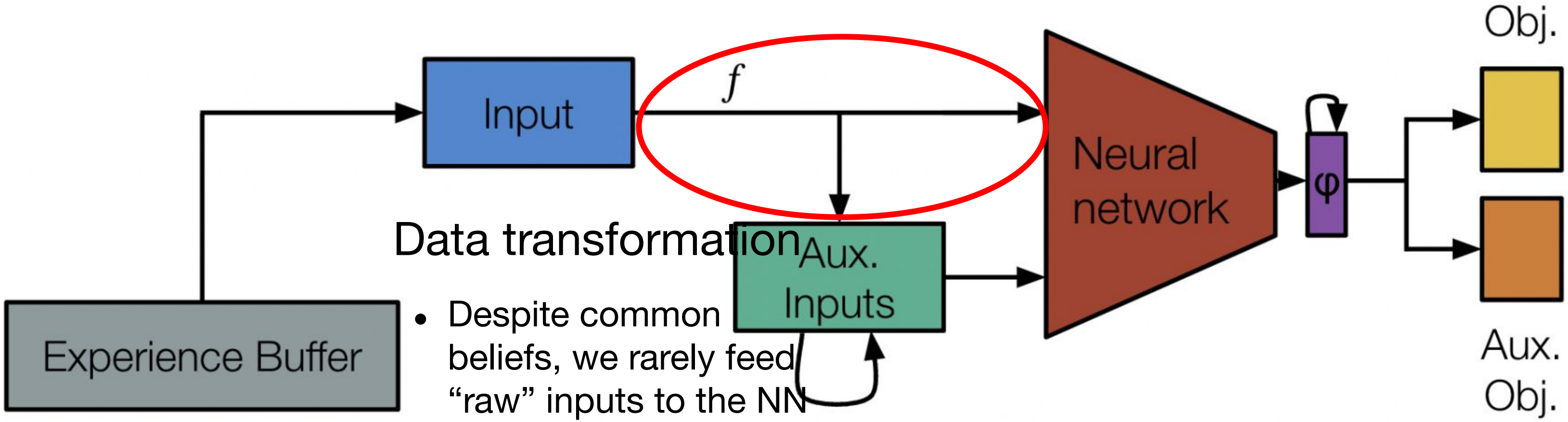
Structural Outline of Model-Free Value-based Deep RL Algs.



Data sampling strategy

- Can decorrelate samples
- Allows for parallelism (mini-batch training) and for samples being used more than once
- Often requires off-policy learning
- Often, most recent sample is not processed right away

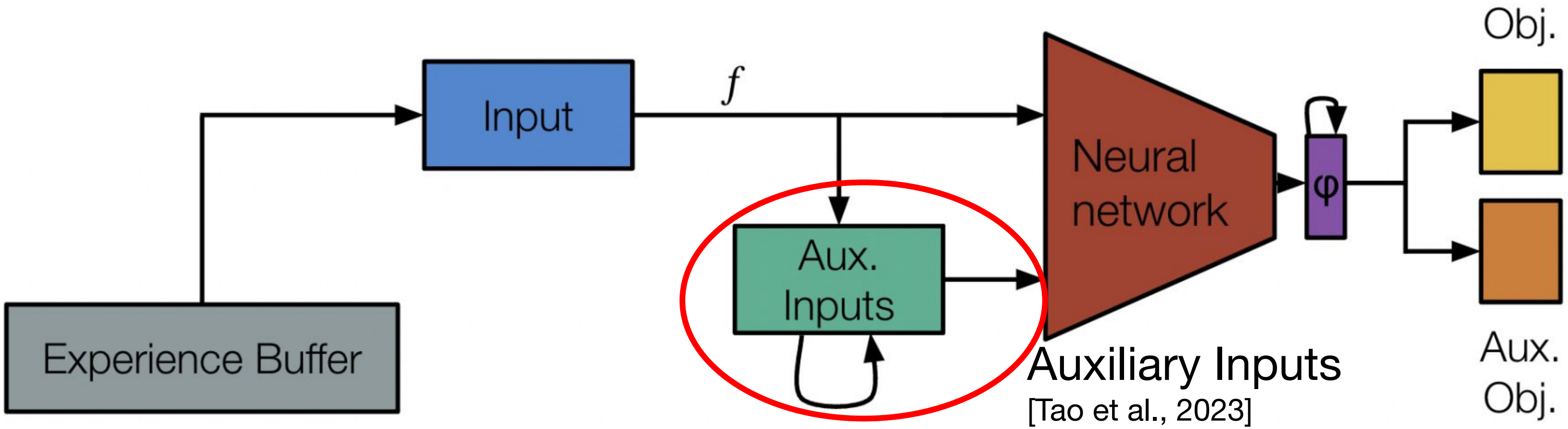
Structural Outline of Model-Free Value-based Deep RL Algs.



Data transformation

- Despite common beliefs, we rarely feed "raw" inputs to the NN
- For images, people sometimes grayscale, downsample, crop
- For proprioception, people can normalize, apply the Fourier transform, and so on

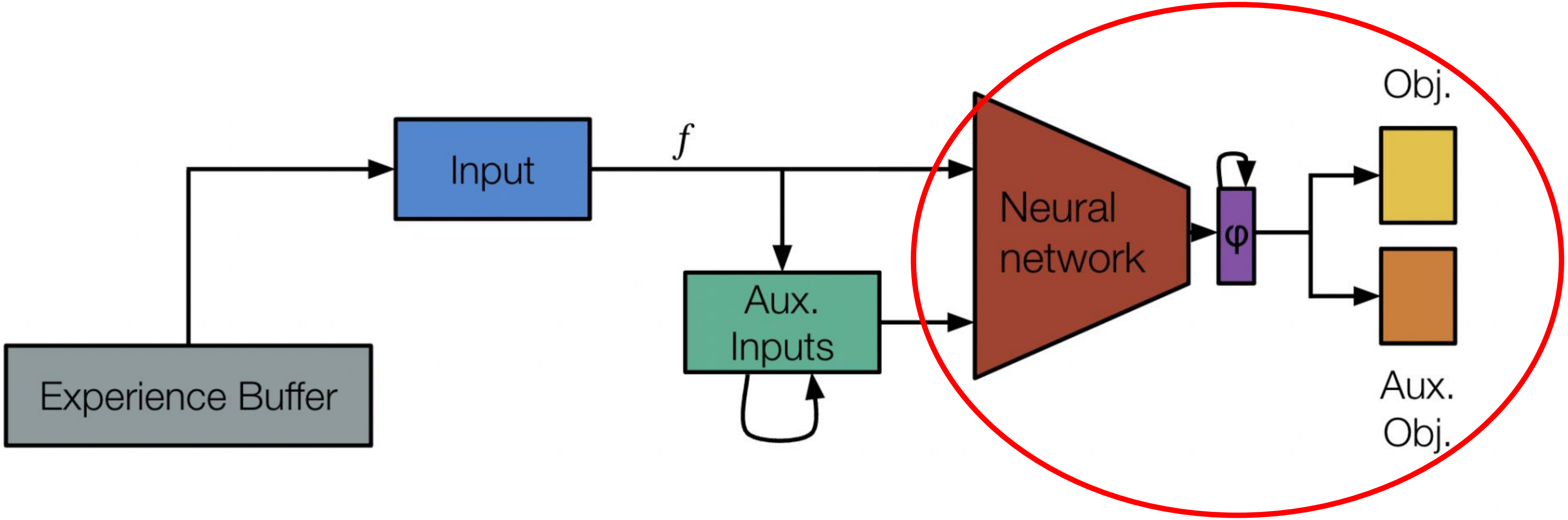
Structural Outline of Model-Free Value-based Deep RL Algs.



Auxiliary Inputs
[Tao et al., 2023]

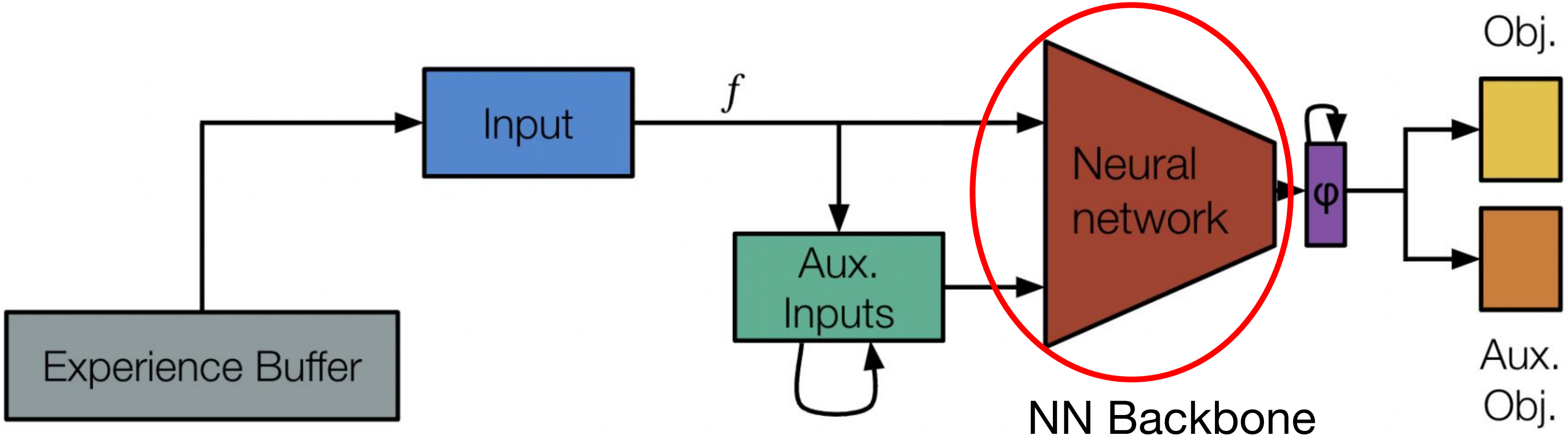
- In most of the high-profile results, the NN received more than the *last* obs. transformed
- Examples include: last *k* frames, uncertainty estimates, predictions

Structural Outline of Model-Free Value-based Deep RL Algs.



Neural Network

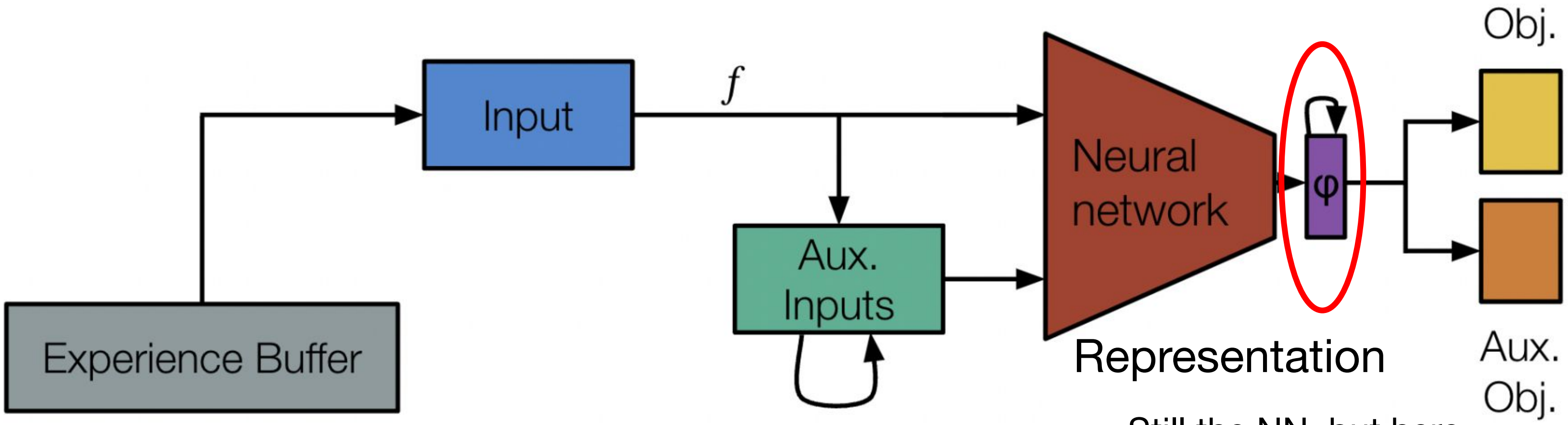
Structural Outline of Model-Free Value-based Deep RL Algs.



NN Backbone

- First couple of layers that will lead to the learned features, ϕ
- Examples include MLPs, CNNs, and ResNets

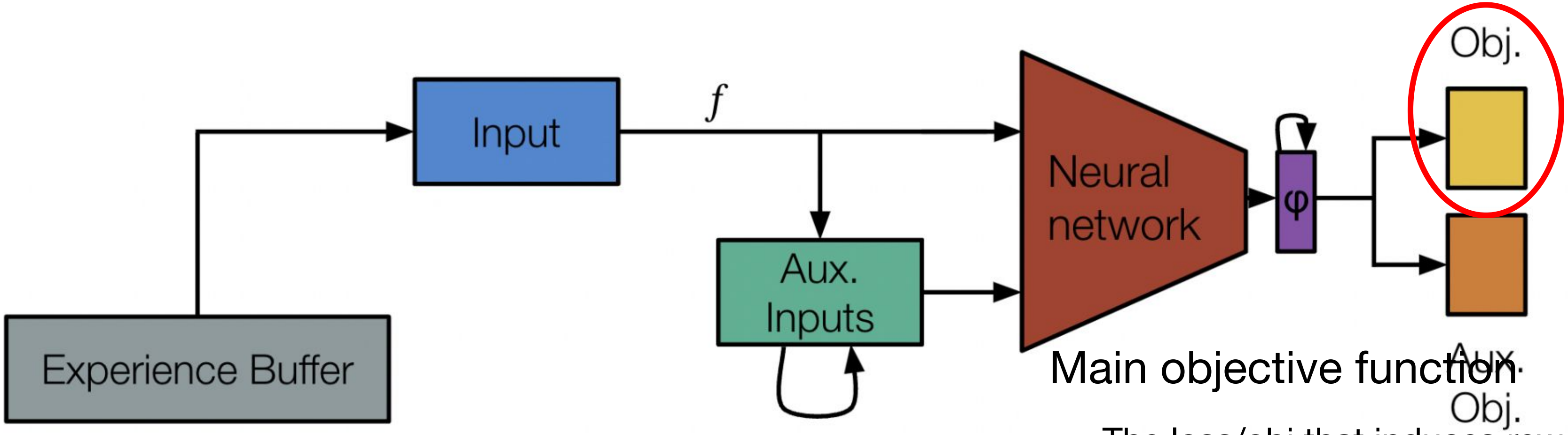
Structural Outline of Model-Free Value-based Deep RL Algs.



Representation

- Still the NN, but here we depict the representation
- It can be seen as the agent-state
- It can be recurrent, but training gets trickier because of the order in which samples are processed

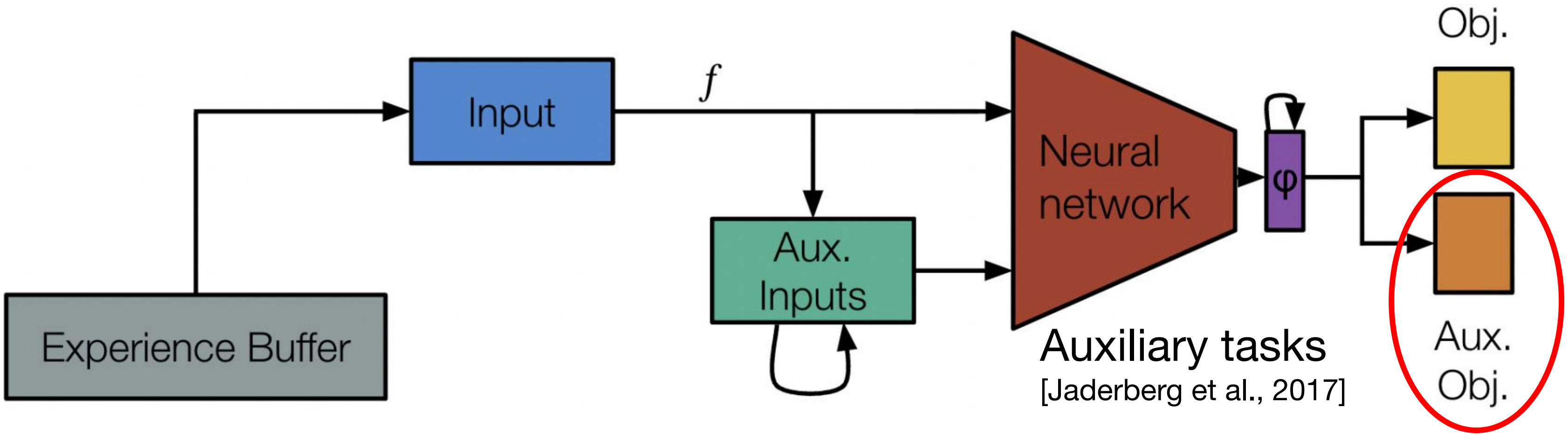
Structural Outline of Model-Free Value-based Deep RL Algs.



Main objective function

- The loss/obj that induces reward maximization (value estimation), used for action selection
- It is often some TD-like obj. func.
- I'm conflating loss and obj. here

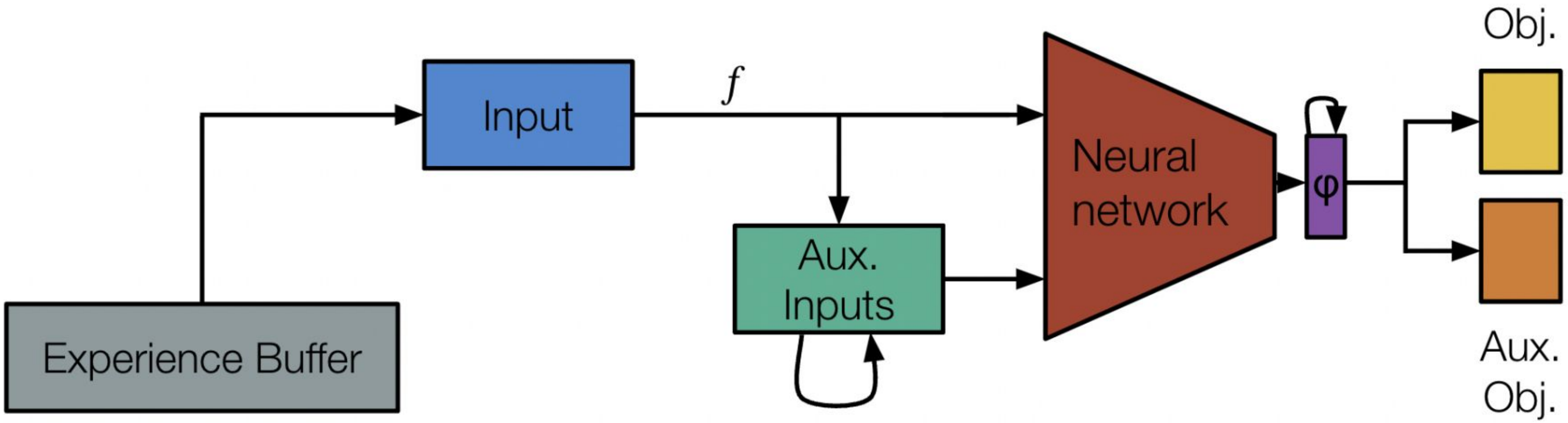
Structural Outline of Model-Free Value-based Deep RL Algs.



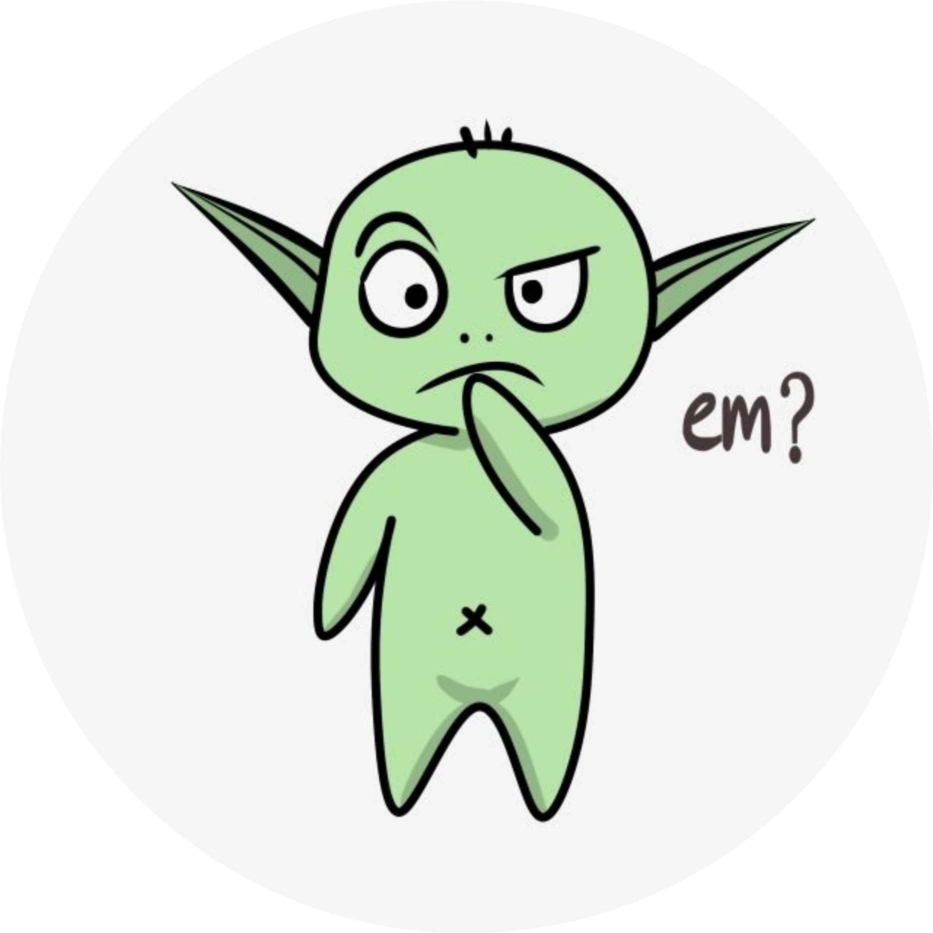
Auxiliary tasks
[Jaderberg et al., 2017]

- Predicting more than the value estimate can be quite helpful when training the NN (GVFs)
- Changes the opt. landscape
- Learn better representations

Structural Outline of Model-Free Value-based Deep RL Algs.

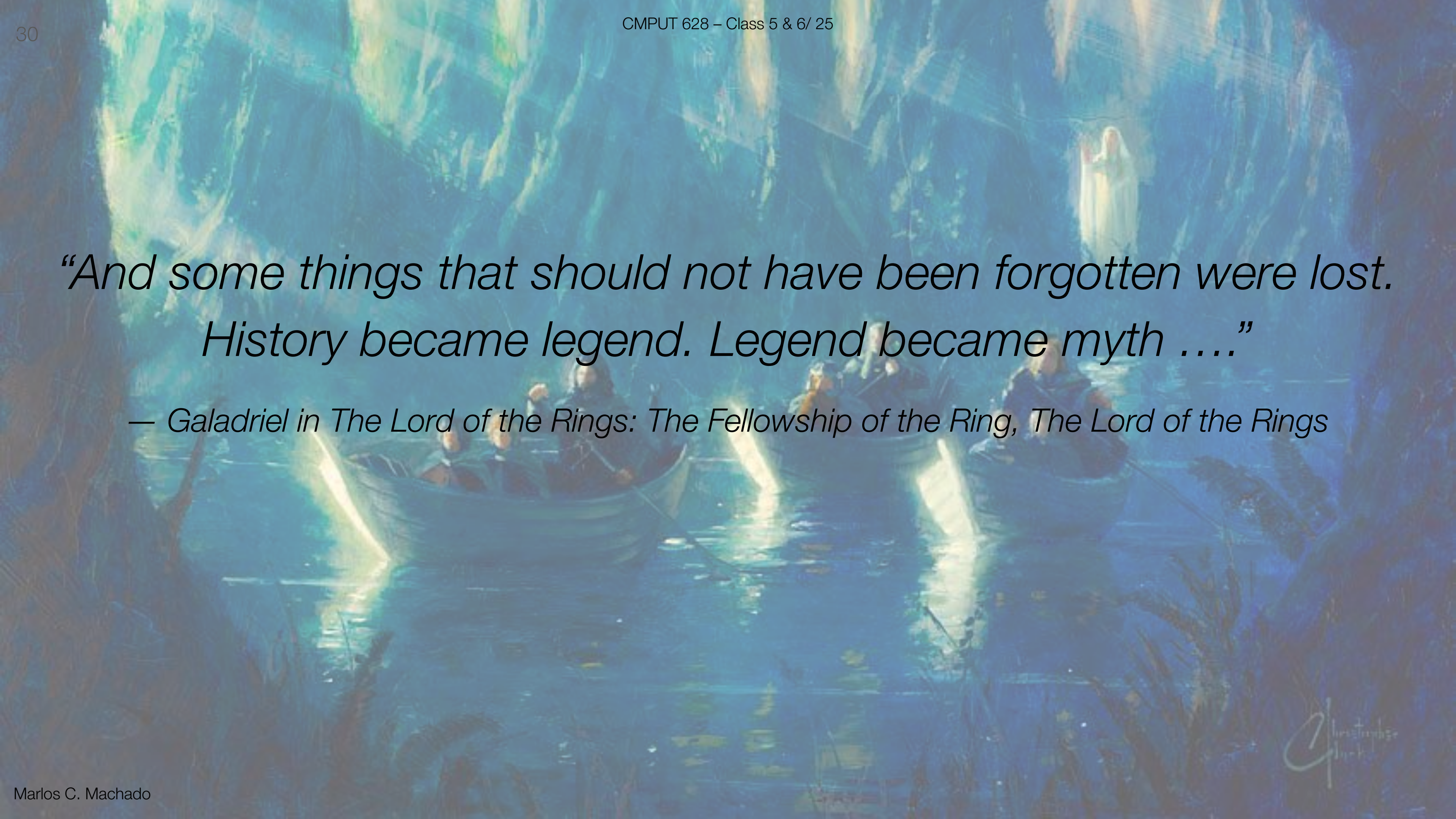


Different algorithms are different instantiations of these boxes.
This will be the major bulk of our course!



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Why use games as an evaluation platform?



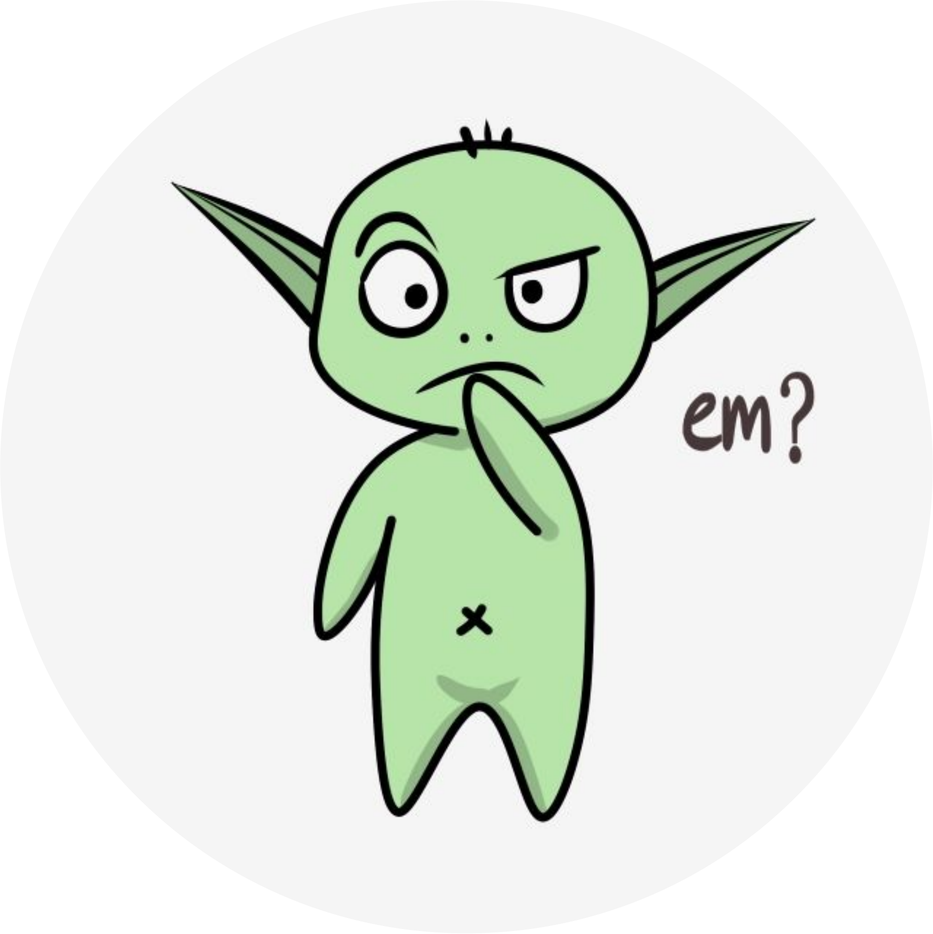
“And some things that should not have been forgotten were lost. History became legend. Legend became myth ….”

— Galadriel in The Lord of the Rings: The Fellowship of the Ring, The Lord of the Rings

Why do we use games as an evaluation platform?

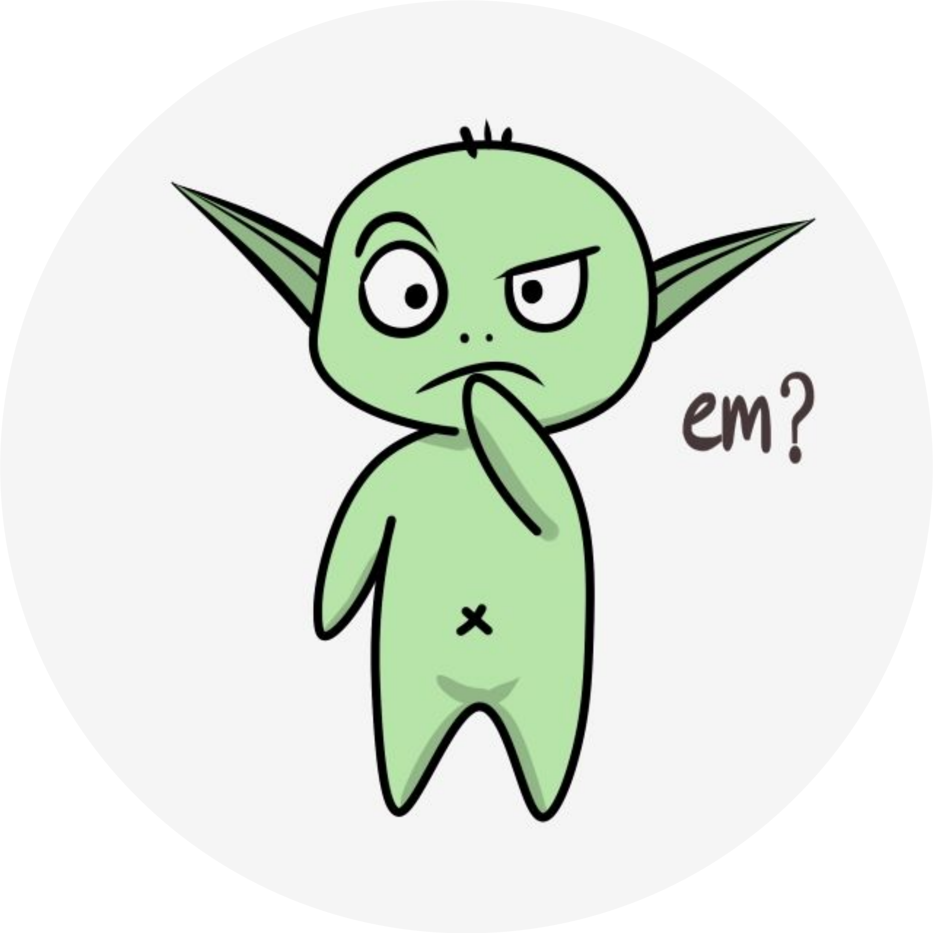
- Games have many useful properties for scientific experimentation:
    - They are fully controllable and have well-defined rules
    - They are free of experimenter's bias
    - They are relatable and challenging
    - They have well-defined metrics of success
- Games require increasingly complex sets of skills that are generally useful:
    - Colours, numbers, pattern matching
    - Effects of actions, short-term planning, long-term planning
    - Strategic thinking, problem solving
    - Cooperation, social skills
- Games are convenient because experts already developed them for us

`Influenced by many discussions with Michael Bowlin`



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

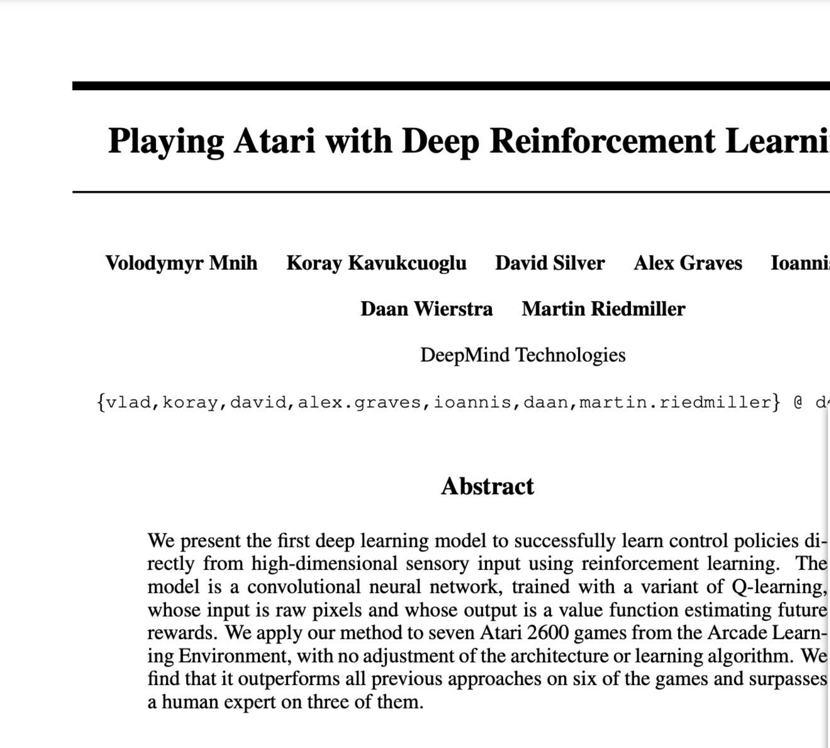
Atari 2600 Games



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Deep Q-Networks (DQN)

The beginning of it all: Mnih et al. (2013)



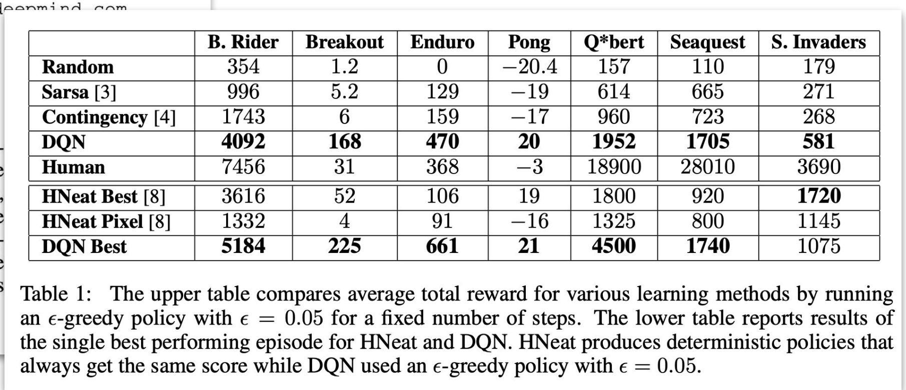
Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou
Daan Wierstra Martin Riedmiller
DeepMind Technologies
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

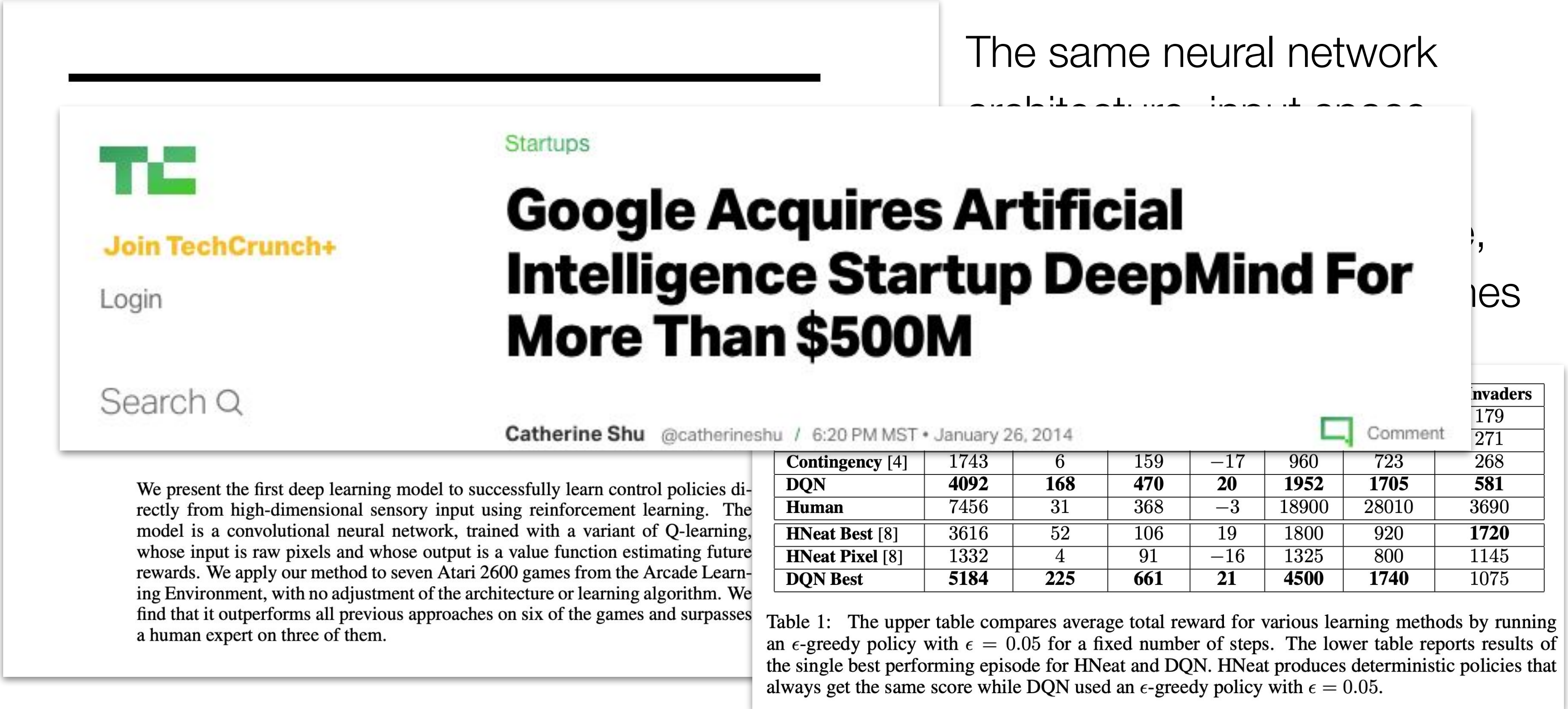
The same neural network architecture, input space, hyperparameters, and, to some degree, action space, were shared across all games



	B. Rider	**Breakout**	**Enduro**	**Pong**	**Q*bert**	**Seaquest**	**S. Invaders**
Random	354	1.2	0	-20.4	157	110	179
Sarsa [3]	996	5.2	129	-19	614	665	271
Contingency [4]	1743	6	159	-17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	-3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	-16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Table 1: The upper table compares average total reward for various learning methods by running an ϵ-greedy policy with $\epsilon = 0.05$ for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an ϵ-greedy policy with $\epsilon = 0.05$.

The beginning of it all: Mnih et al. (2013)



Startups

Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Catherine Shu @catherineshu / 6:20 PM MST • January 26, 2014

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

							nvaders
							179
							271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Table 1: The upper table compares average total reward for various learning methods by running an ϵ-greedy policy with $\epsilon = 0.05$ for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an ϵ-greedy policy with $\epsilon = 0.05$.

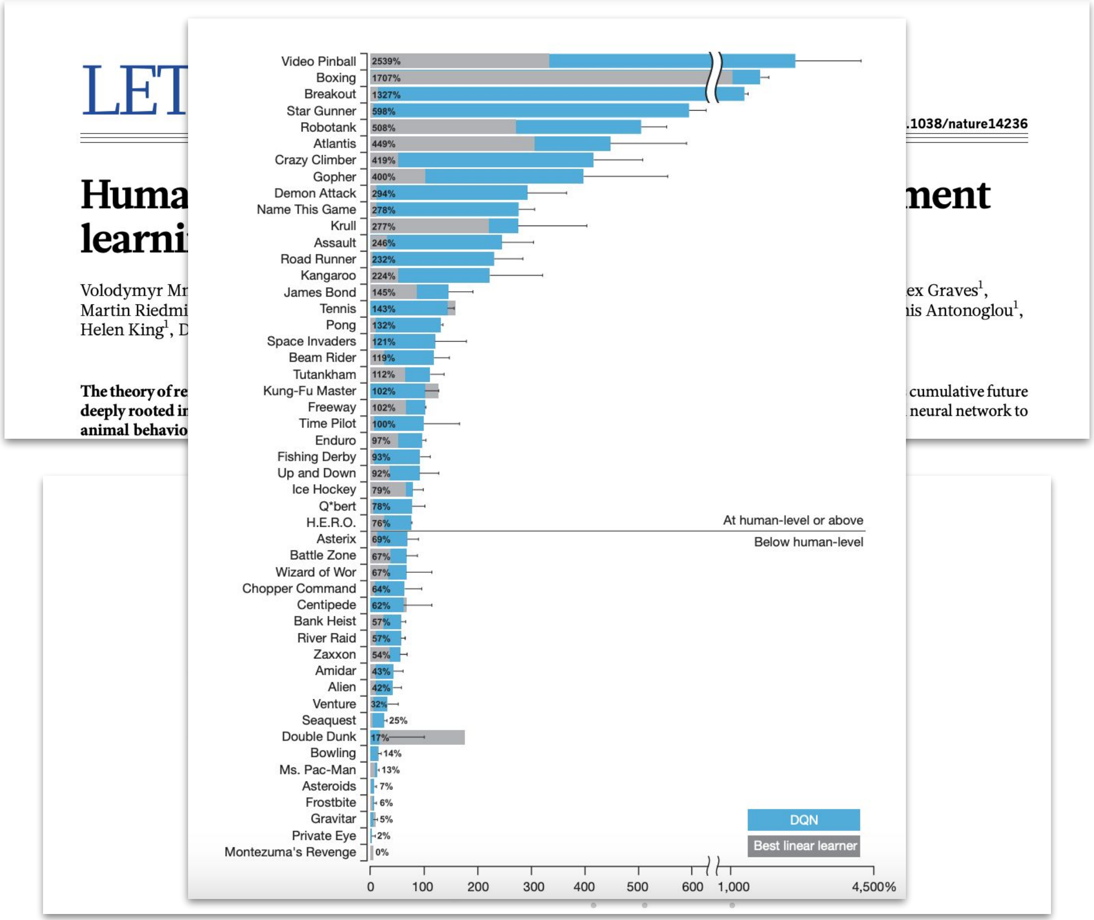
The same neural network
architecture, input space

e,

es

… and finally

INNOVATIONS IN
The microbiome
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LEARNING CURVE
Self-taught AI software attains human-level performance in video games
PAGES 486 & 529
EPIDEMIOLOGY
SHARE DATA IN OUTBREAKS
Forge open access to sequences and more
PAGE 477
COSMOLOGY
A GIANT IN THE EARLY UNIVERSE
A supermassive black hole at a redshift of 6.3
PAGES 490 & 512
QUANTUM PHYSICS
TELEPORTATION FOR TWO
Transferring two properties of a single photon
PAGES 491 & 516
NATURE.COM/NATURE
26 February 2015 £10
Vol. 518, No. 7540



LET

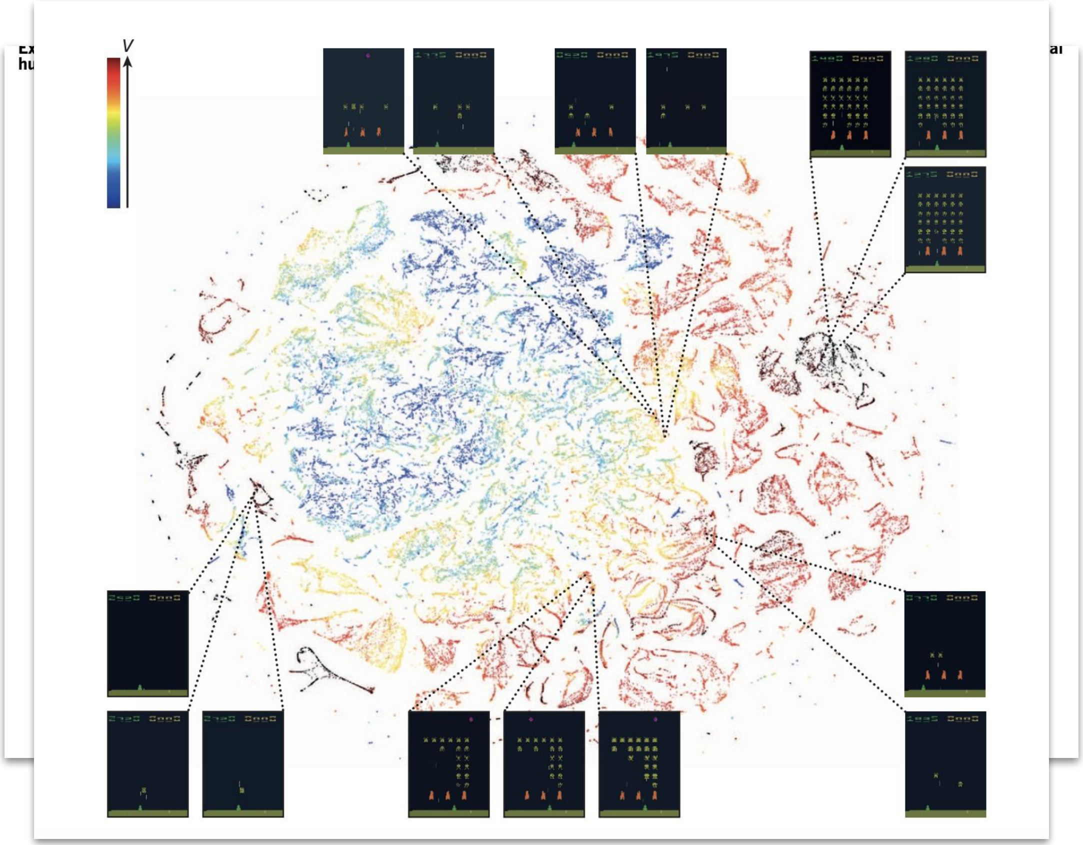
.1038/nature14236

Huma … ment
learni

Volodymyr Mn … ex Graves[1],
Martin Riedmi … nis Antonoglou[1],
Helen King[1], D

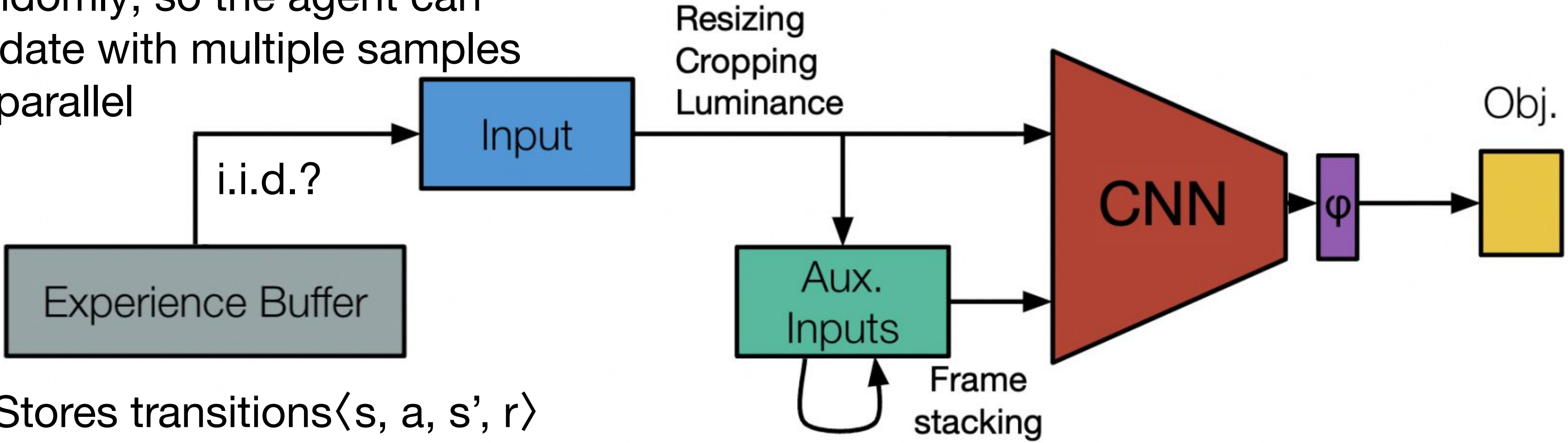
The theory of re … cumulative future
deeply rooted in … neural network to
animal behavio

And the results caught everyone’s attention



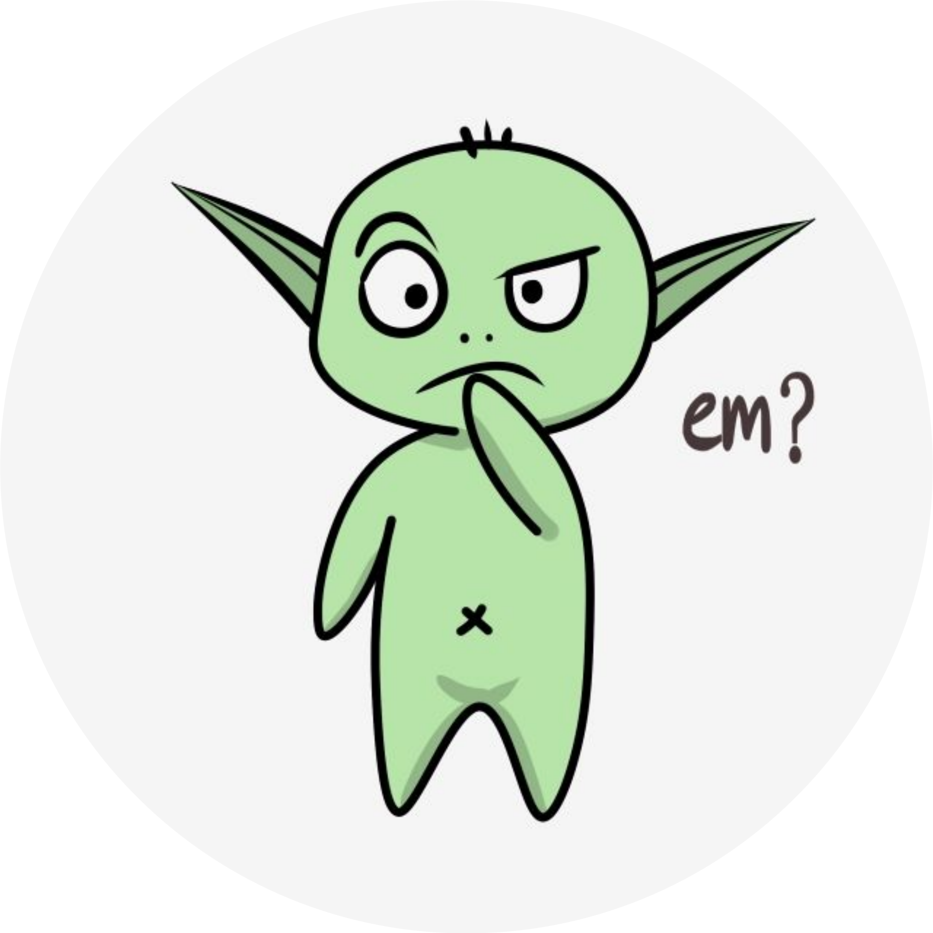
DQN: A first complete instantiation of our structural outline

Transitions are sampled randomly, so the agent can update with multiple samples in parallel



Stores transitions⟨s, a, s’, r⟩ into a FIFO queue. They were generated by many policies.

Buffer size: 1M
Minibatch size: 32
Update frequency: 4



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

DQN objective function and loss function I

$$\mathcal{L}^{\text{DQN}} = \mathbb{E}_{(o,a,r,o')\sim U(\mathcal{D})}\left[\left(R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(O_{t+1}, a'; \boldsymbol{\theta}^{-}) - Q(O_t, A_t; \boldsymbol{\theta}_t)\right)^2\right]$$

Expectation is over the uniform distribution of sampled transitions

Target network

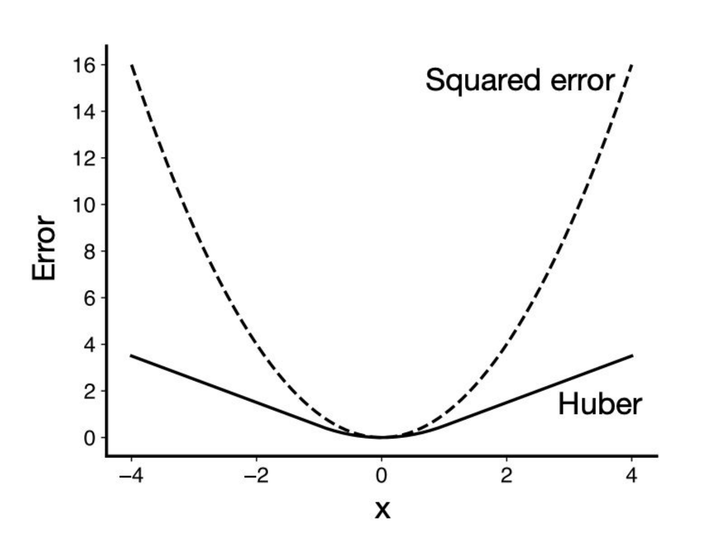
$$Y(R_{t+1}, O_{t+1}; \boldsymbol{\theta}^{-}) = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q\big(O_{t+1}, a'; \boldsymbol{\theta}^{-}\big)$$

$$\mathcal{L}^{\text{DQN}} = \mathbb{E}_{(o,a,r,o')\sim U(\mathcal{D})}\left[\left(Y(R_{t+1}, O_{t+1}; \boldsymbol{\theta}^{-}) - Q(O_t, A_t; \boldsymbol{\theta}_t)\right)^2\right].$$

Regression

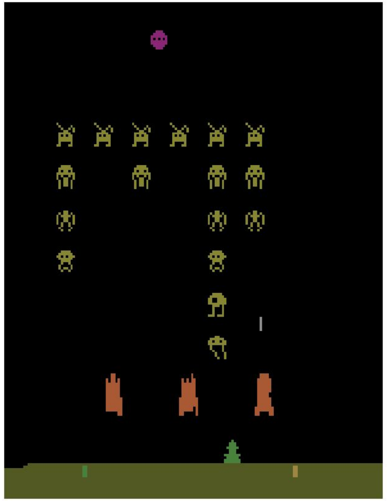
DQN objective function and loss function II

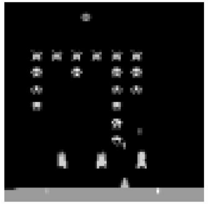
- DQN, at least Nature-DQN, had way more than an experience replay buffer and a target network
  - To use the same hyperparameters across all the games, the TD error and the loss function had to be in the same scale:
    - Clip the rewards to be [-1, 0, 1]
    - Clip the TD error be between –1 and 1 in the gradient update
    - RMSProp (Adam is now more common)
  - The experience replay buffer needs to be partially filled before training can occur
    - Uniform random policy for 50k frames
    - ϵ-greedy policy, with ϵ decaying from 1.0 to 0.1 within the first 1M frames

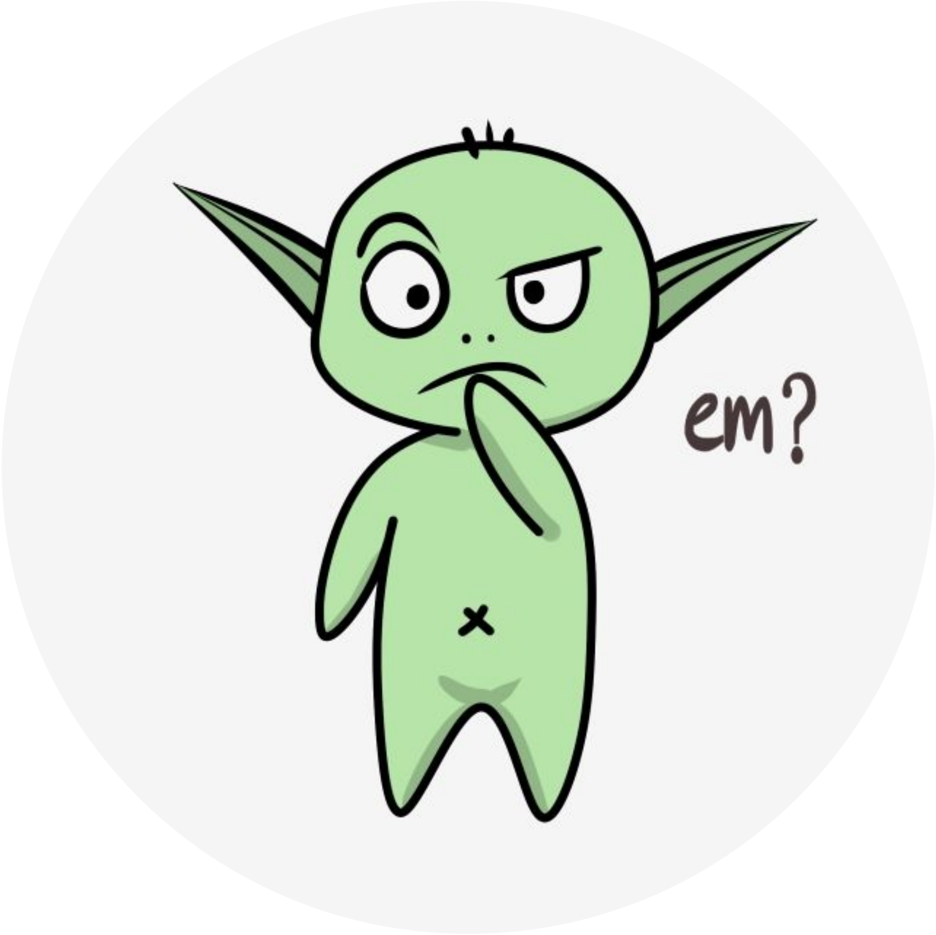


More design choices for DQN

- Observations: Four stacked frames in an attempt to make it Markovian
  I call the additional three frames *auxiliary inputs* (Tao et al., 2023)
  - Instead of 210 × 160 pixels with 128 colours, the observation is pre-processed. It is rescaled and recoloured to an image of size 84 × 84 where the luminance values are extracted from the original image and used to encode each pixel (grayscaling).
- Neural network architecture
  - 3 convolutional layers, a hidden layer, and an output layer
    Conv. layers use 32 (8 × 8), 64 (4 × 4), and 64 (3 × 3) filters with stride 4, 2, and 1 respectively. These layers are flattened (3,136 units) to connect to a hidden layer of 512 units, which is connected to the output layer.
    The output layer size ranges from 4 to 18 units.
  - ReLus everywhere.







https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

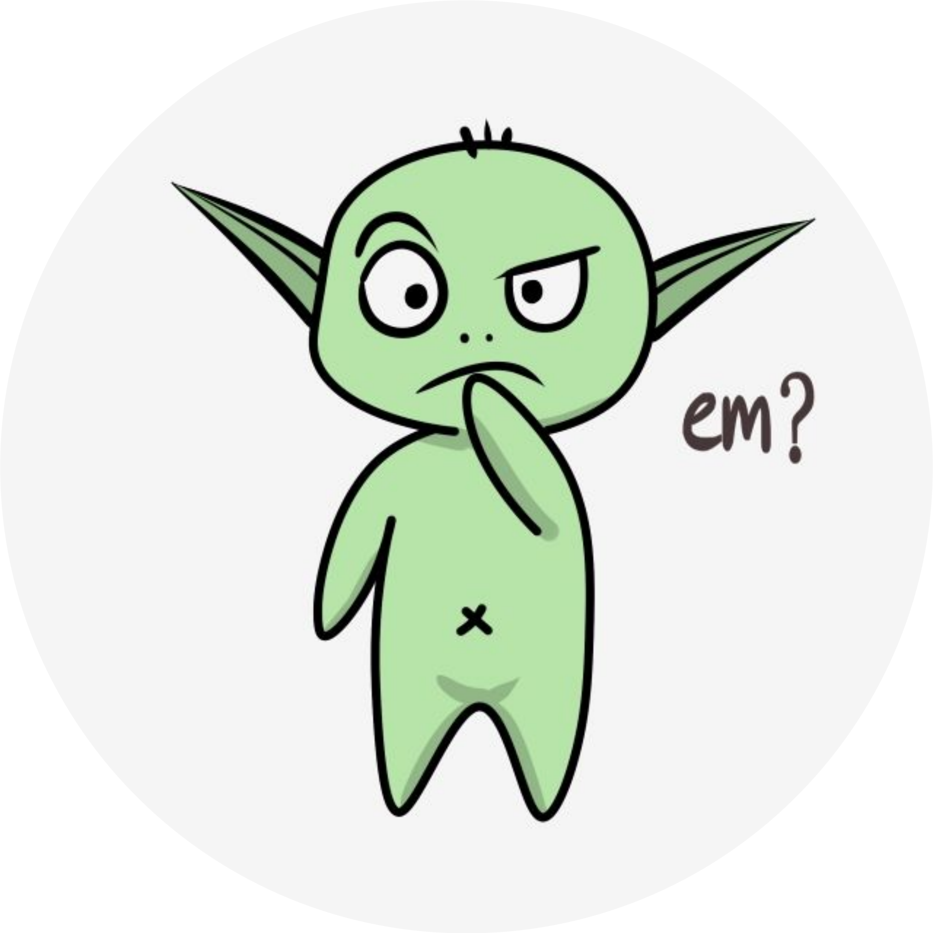
DQN and Neural Fitted-Q Iteration

- Neural Fitted Q-Iteration (NFQI) is DQN’s predecessor, *they are not the same*.
- Neural Fitted Q-Iteration (NFQI):
  - It trains a neural network from scratch in each iteration.
  - It keeps all the data it has seen around (the most natural instantiation of the algorithm collects all data beforehand). Thus the intermediate Q-values do not impact the transitions observed by the agent (although the authors do mention a variant in which the dataset is augmented with more data).
  - It has some sort of target network as the target values the neural network is regressing to are fixed to the values of the previous iteration, implying the frequency in which the target network is updated is at least the number of samples in the collected dataset.
  - It was evaluated only in classic simple tasks such as Mountain Car and Pole Balancing.

Target networks and experience replay buffers aren’t *hacks*

- Replay buffers give us more i.i.d. samples, sample reuse and parallelization
  - It can be seen as a model. Dyna was always praised, why the issue with a replay buffer?
  - The universe doesn’t shuffle data, though
  - I find it weird to not perform updates with the sample the agent just saw (but it often doesn’t help)
  - The size of the replay buffer is a key (and very sensitive) hyperparameter in deep RL agents
- Target networks approximate the fixed targets in supervised learning
  - Neural Fitted-Q Iteration and other methods did that before
  - But they can slow down learning because the agent is always regressing towards a stale value
  - The rate at which the target network is updated is a rather sensitive hyperparameter

Replay ratio: the number of environment steps taken per gradient step.



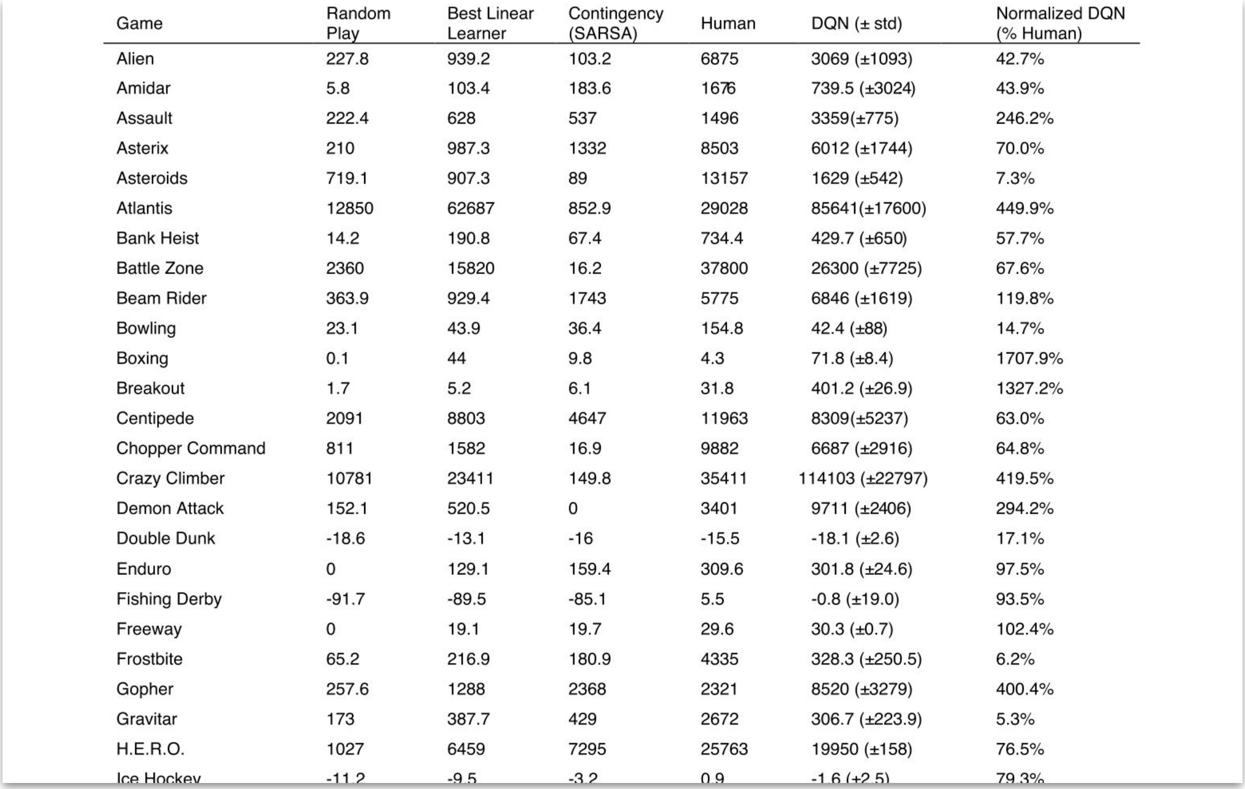
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

DQN’s evaluation methodology

- Final performance was reported after training an agent with 200 million frames
    - Each action was repeated by the agent 4 times, this parameter is known as *frame skip*
    - Four actions were selected by the agent between successive updates
- Episode terminated upon loss of a life
- Minimal action set was used
- There was an evaluation phase in which the performance of the *best* checkpoint obtained during learning was evaluated 30 times. Learning happened once
- The ALE used to be deterministic, so they used 0-30 no-ops to provide *some* randomization

○ 200M frames
○ 50M action selections
○ 12.5M grad. updates

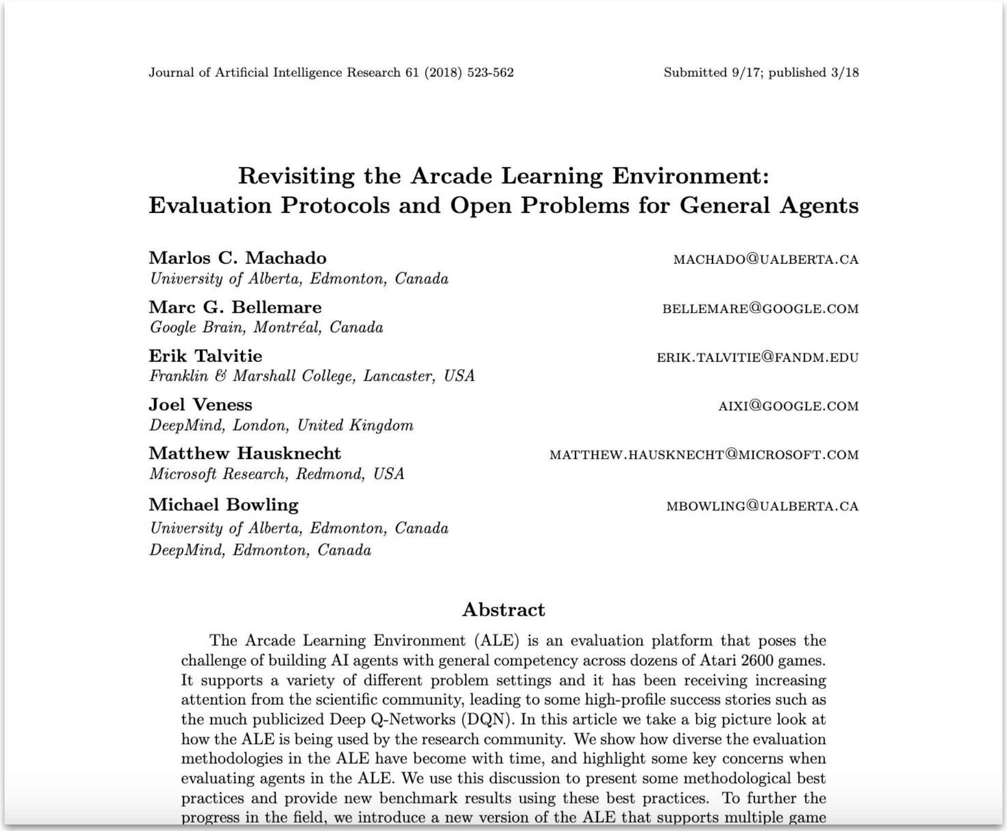
On the comparison between different papers



Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	227.8	939.2	103.2	6875	3069 (±1093)	42.7%
Amidar	5.8	103.4	183.6	1676	739.5 (±3024)	43.9%
Assault	222.4	628	537	1496	3359(±775)	246.2%
Asterix	210	987.3	1332	8503	6012 (±1744)	70.0%
Asteroids	719.1	907.3	89	13157	1629 (±542)	7.3%
Atlantis	12850	62687	852.9	29028	85641(±17600)	449.9%
Bank Heist	14.2	190.8	67.4	734.4	429.7 (±650)	57.7%
Battle Zone	2360	15820	16.2	37800	26300 (±7725)	67.6%
Beam Rider	363.9	929.4	1743	5775	6846 (±1619)	119.8%
Bowling	23.1	43.9	36.4	154.8	42.4 (±88)	14.7%
Boxing	0.1	44	9.8	4.3	71.8 (±8.4)	1707.9%
Breakout	1.7	5.2	6.1	31.8	401.2 (±26.9)	1327.2%
Centipede	2091	8803	4647	11963	8309(±5237)	63.0%
Chopper Command	811	1582	16.9	9882	6687 (±2916)	64.8%
Crazy Climber	10781	23411	149.8	35411	114103 (±22797)	419.5%
Demon Attack	152.1	520.5	0	3401	9711 (±2406)	294.2%
Double Dunk	-18.6	-13.1	-16	-15.5	-18.1 (±2.6)	17.1%
Enduro	0	129.1	159.4	309.6	301.8 (±24.6)	97.5%
Fishing Derby	-91.7	-89.5	-85.1	5.5	-0.8 (±19.0)	93.5%
Freeway	0	19.1	19.7	29.6	30.3 (±0.7)	102.4%
Frostbite	65.2	216.9	180.9	4335	328.3 (±250.5)	6.2%
Gopher	257.6	1288	2368	2321	8520 (±3279)	400.4%
Gravitar	173	387.7	429	2672	306.7 (±223.9)	5.3%
H.E.R.O.	1027	6459	7295	25763	19950 (±158)	76.5%
Ice Hockey	-11.2	-9.5	-3.2	0.9	-1.6 (±2.5)	79.3%

We always report the game score, because that’s what we care about. We often set γ < 1, though. It is a better surrogate objective due to instabilities when setting γ = 1.

Standardizing experimentation and introducing stochasticity



Journal of Artificial Intelligence Research 61 (2018) 523-562 Submitted 9/17; published 3/18

Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents

Marlos C. Machado MACHADO@UALBERTA.CA
University of Alberta, Edmonton, Canada

Marc G. Bellemare BELLEMARE@GOOGLE.COM
Google Brain, Montréal, Canada

Erik Talvitie ERIK.TALVITIE@FANDM.EDU
Franklin & Marshall College, Lancaster, USA

Joel Veness AIXI@GOOGLE.COM
DeepMind, London, United Kingdom

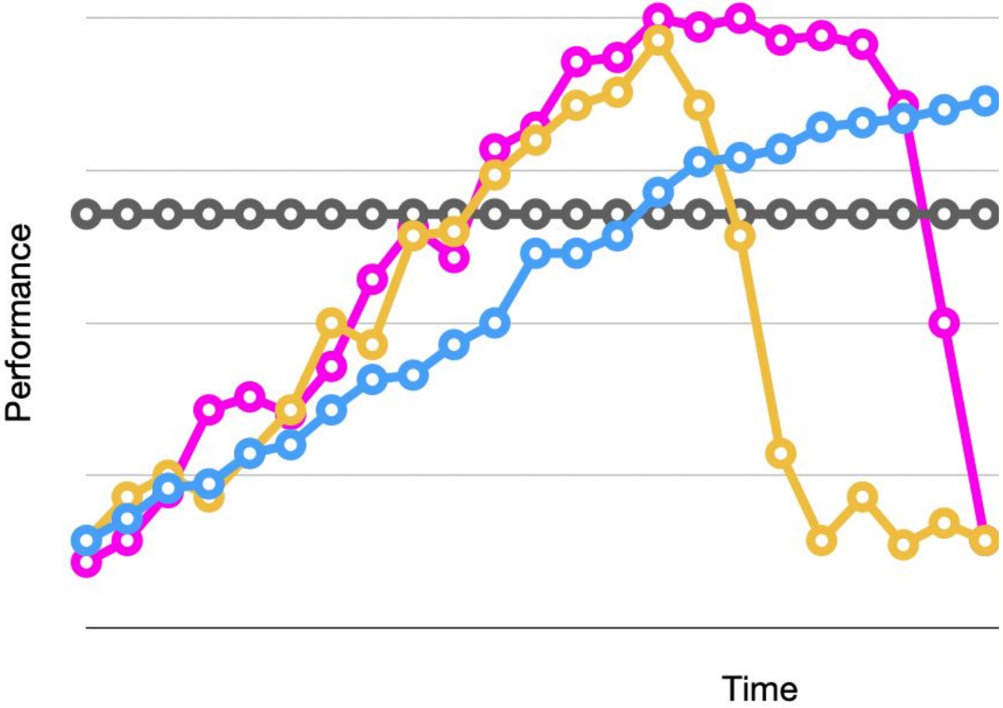
Matthew Hausknecht MATTHEW.HAUSKNECHT@MICROSOFT.COM
Microsoft Research, Redmond, USA

Michael Bowling MBOWLING@UALBERTA.CA
University of Alberta, Edmonton, Canada
DeepMind, Edmonton, Canada

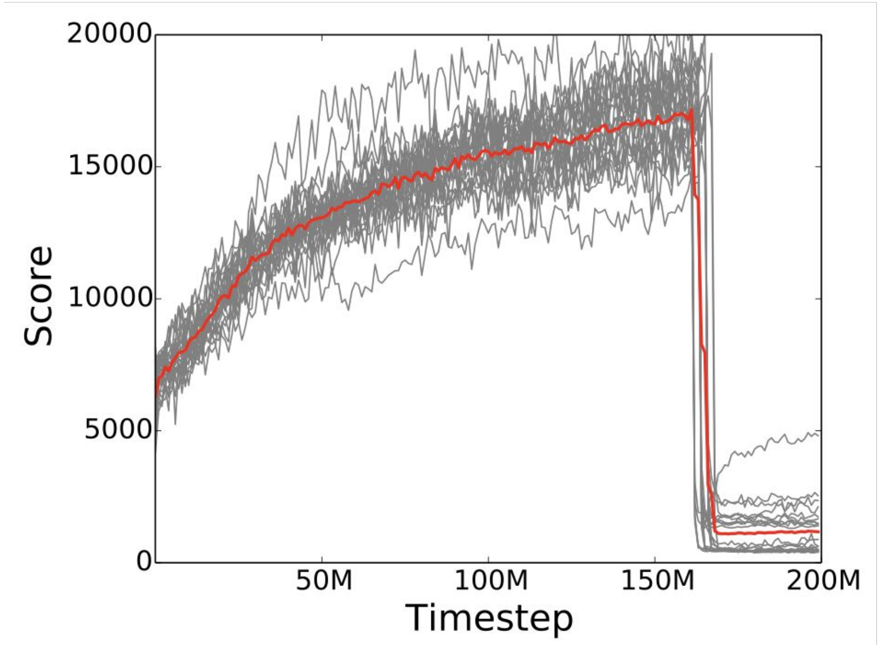
Abstract

The Arcade Learning Environment (ALE) is an evaluation platform that poses the challenge of building AI agents with general competency across dozens of Atari 2600 games. It supports a variety of different problem settings and it has been receiving increasing attention from the scientific community, leading to some high-profile success stories such as the much publicized Deep Q-Networks (DQN). In this article we take a big picture look at how the ALE is being used by the research community. We show how diverse the evaluation methodologies in the ALE have become with time, and highlight some key concerns when evaluating agents in the ALE. We use this discussion to present some methodological best practices and provide new benchmark results using these best practices. To further the progress in the field, we introduce a new version of the ALE that supports multiple game

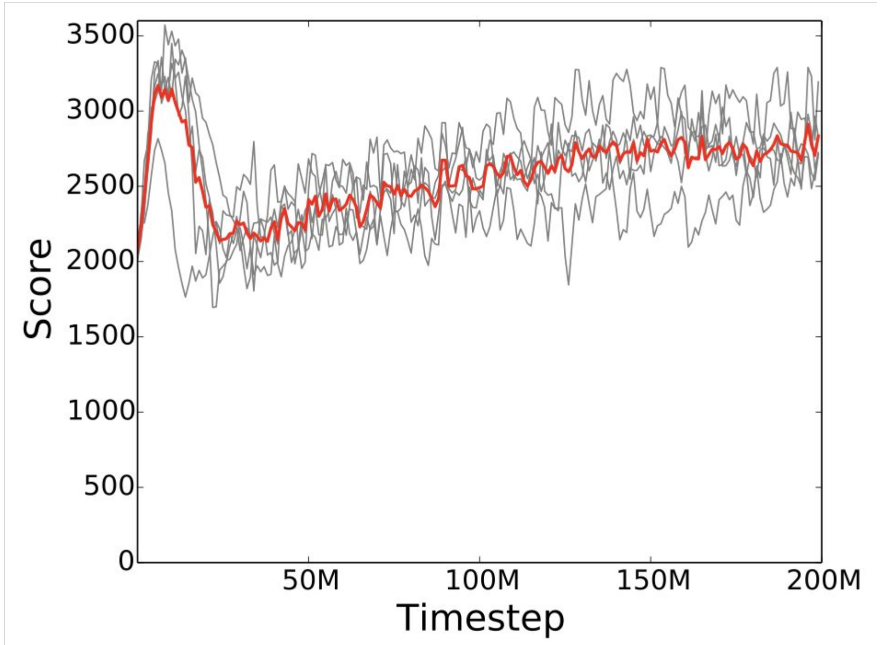
Which one is better?



It does happen!



Sarsa (λ)

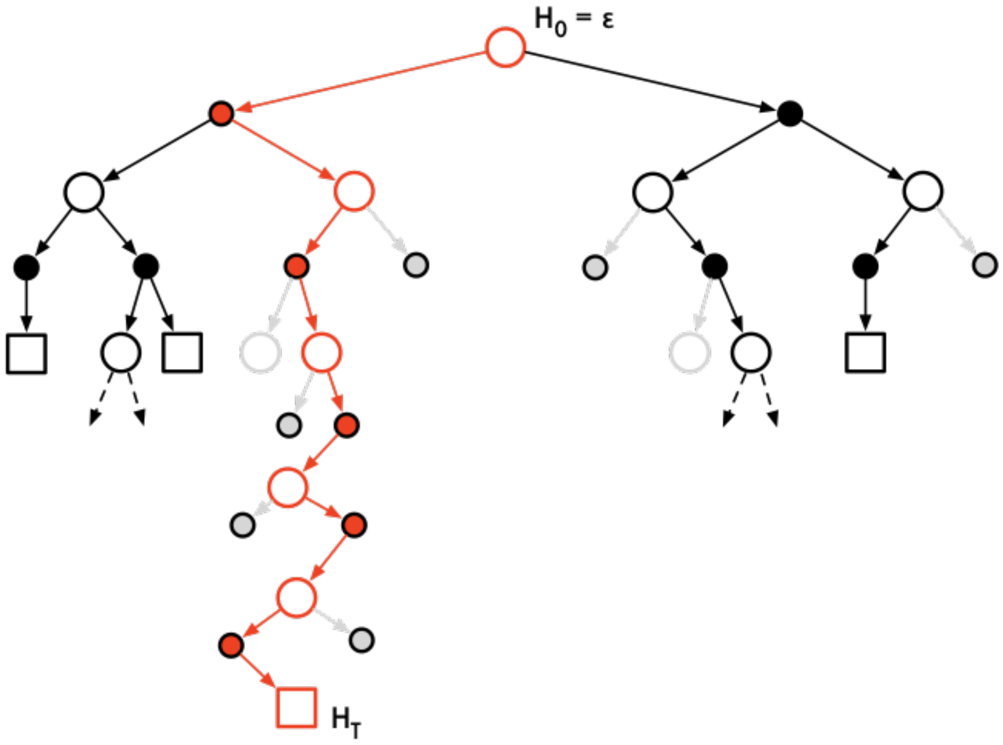


DQN

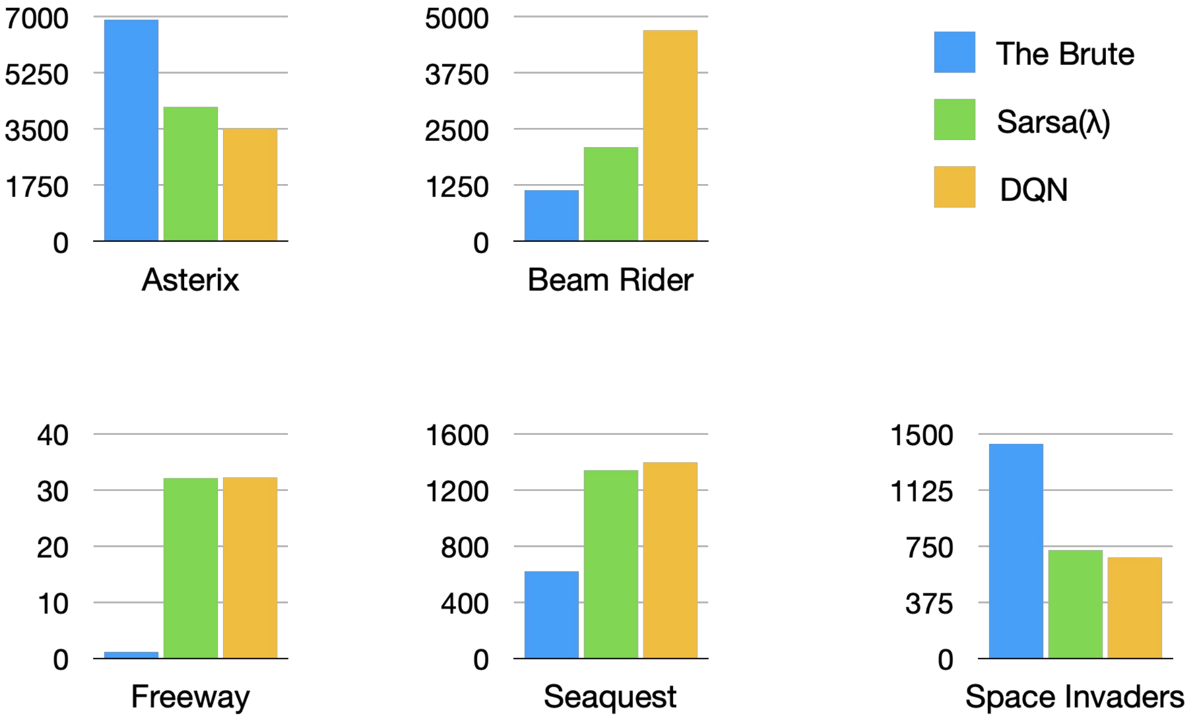
Determinism in the Arcade Learning Environment

- The ALE was deterministic:
  - all episodes have the same start state and,
  - the same sequence of actions will always lead to the same outcome.

The Brute [Bellemare et al., 2015]

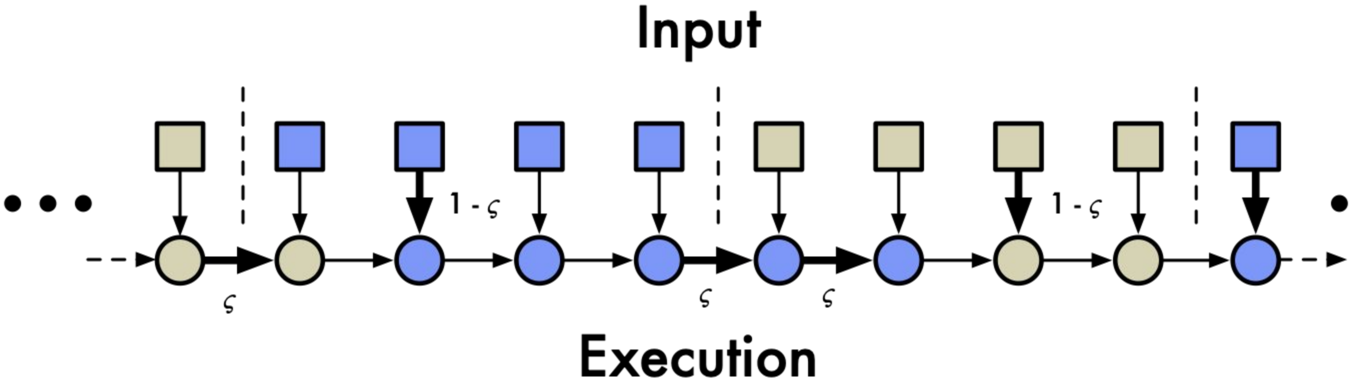


Performance in the deterministic ALE

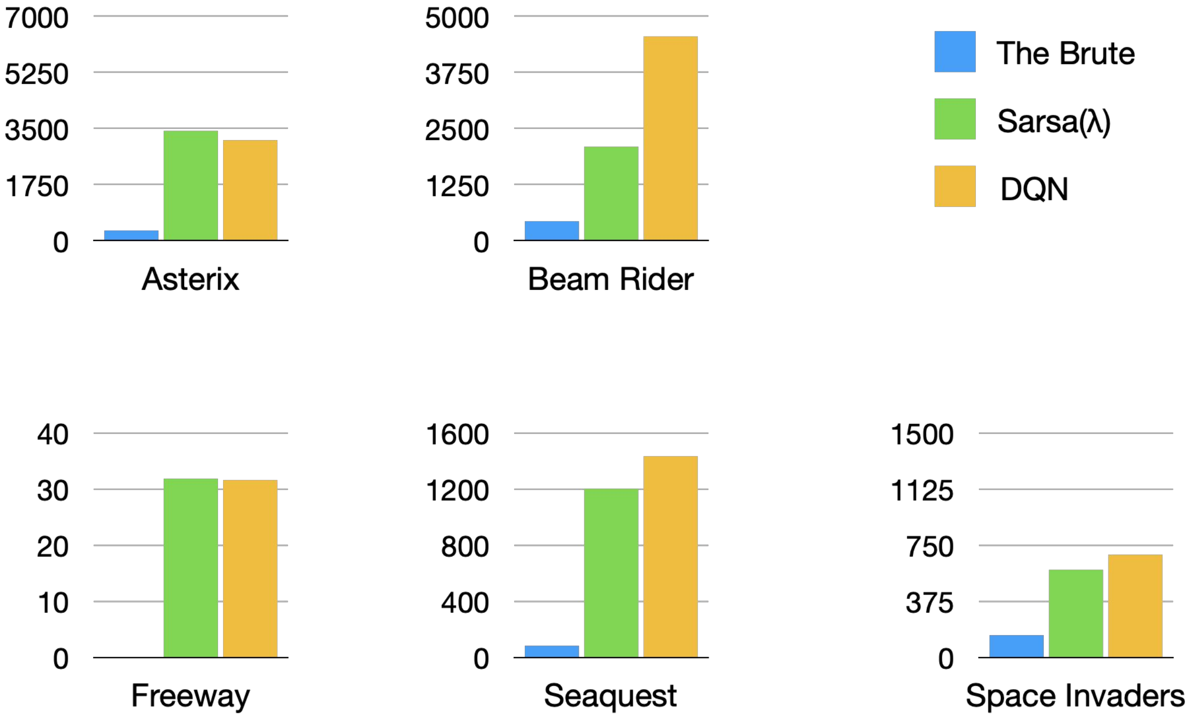


Stochasticity model – *Sticky actions*

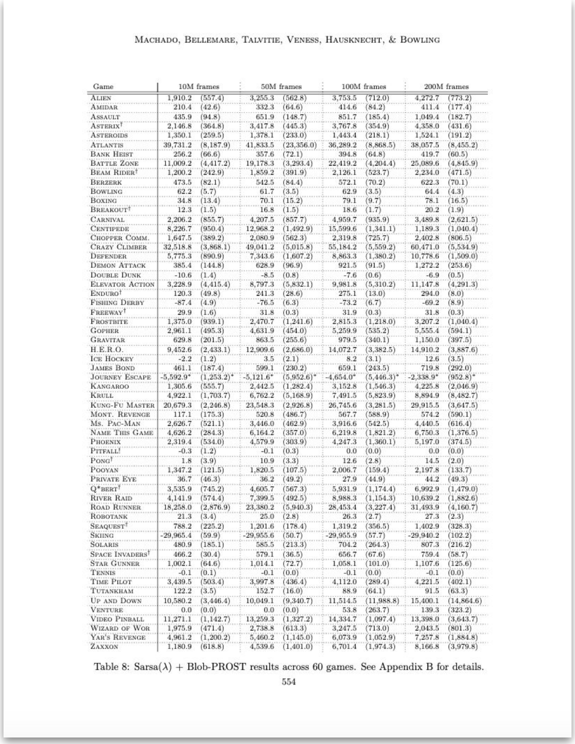
$$A_t = \begin{cases} a, & \text{with prob.} \quad 1 - \varsigma, \\ a_{t-1}, & \text{with prob.} \quad \varsigma. \end{cases}$$



Performance in the stochastic ALE



Different RL methods were comparable again. We did not even emphasize benchmarking too much (it didn't matter)

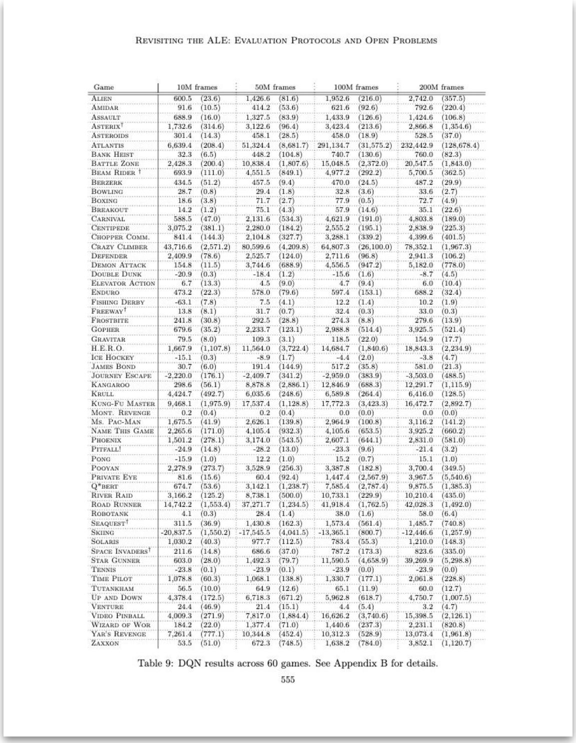


Machado, Bellemare, Talvitie, Veness, Hausknecht, & Bowling

Game	10M frames		50M frames		100M frames		200M frames	
Alien	1,910.2	(557.4)	3,255.3	(562.8)	3,753.5	(712.0)	4,272.7	(773.2)
Amidar	210.4	(42.6)	332.3	(64.6)	414.6	(84.2)	411.4	(177.4)
Assault	435.9	(94.8)	651.9	(148.7)	851.7	(185.4)	1,049.4	(182.7)
Asterix†	2,146.8	(364.8)	3,417.8	(445.3)	3,767.8	(354.9)	4,358.0	(431.6)
Asteroids	1,350.1	(259.5)	1,378.1	(233.0)	1,443.4	(218.1)	1,524.1	(191.2)
Atlantis	39,731.2	(8,187.9)	41,833.5	(23,356.0)	36,289.2	(8,868.5)	38,057.5	(8,455.2)
Bank Heist	256.2	(66.6)	357.6	(72.1)	394.8	(64.8)	419.7	(60.5)
Battle Zone	11,009.2	(4,417.2)	19,178.3	(3,293.4)	22,419.2	(4,204.4)	25,089.6	(4,845.9)
Beam Rider†	1,200.2	(242.9)	1,859.2	(391.9)	2,126.1	(523.7)	2,234.0	(471.5)
Berzerk	473.5	(82.1)	542.5	(84.4)	572.1	(70.2)	622.3	(70.1)
Bowling	62.2	(5.7)	61.7	(3.5)	62.9	(3.5)	64.4	(4.3)
Boxing	34.8	(13.4)	70.1	(15.2)	79.1	(9.7)	78.1	(16.5)
Breakout†	12.3	(1.5)	16.8	(1.5)	18.6	(1.7)	20.2	(1.9)
Carnival	2,206.2	(855.7)	4,207.5	(857.7)	4,959.7	(935.9)	3,489.8	(2,621.5)
Centipede	8,226.7	(950.4)	12,968.2	(1,492.9)	15,599.6	(1,341.1)	1,189.3	(1,040.4)
Chopper Comm.	1,647.5	(389.2)	2,080.9	(562.3)	2,319.8	(725.7)	2,402.8	(806.5)
Crazy Climber	32,518.8	(3,868.1)	49,041.2	(5,015.8)	55,184.2	(5,559.2)	60,471.0	(5,534.9)
Defender	5,775.3	(890.9)	7,343.6	(1,607.2)	8,863.3	(1,380.2)	10,778.6	(1,509.0)
Demon Attack	385.4	(144.8)	628.9	(96.9)	921.5	(91.5)	1,272.2	(253.6)
Double Dunk	-10.6	(1.4)	-8.5	(0.8)	-7.6	(0.6)	-6.9	(0.5)
Elevator Action	3,228.9	(4,415.4)	8,797.3	(5,832.1)	9,981.8	(5,310.2)	11,147.8	(4,291.3)
Enduro†	120.3	(49.8)	241.3	(28.6)	275.1	(13.0)	294.0	(8.0)
Fishing Derby	-87.4	(4.9)	-76.5	(6.3)	-73.2	(6.7)	-69.2	(8.9)
Freeway†	29.9	(1.6)	31.8	(0.3)	31.9	(0.3)	31.8	(0.3)
Frostbite	1,375.0	(939.1)	2,470.7	(1,241.6)	2,815.3	(1,218.0)	3,207.2	(1,040.4)
Gopher	2,961.1	(495.3)	4,631.9	(454.0)	5,259.9	(535.2)	5,555.4	(594.1)
Gravitar	629.8	(201.5)	863.5	(255.6)	979.5	(340.1)	1,150.0	(397.5)
H.E.R.O.	9,452.6	(2,433.1)	12,909.6	(2,686.0)	14,072.7	(3,382.5)	14,910.2	(3,887.6)
Ice Hockey	-2.2	(1.2)	3.5	(2.1)	8.2	(3.1)	12.6	(3.5)
James Bond	461.1	(187.4)	599.1	(230.2)	659.1	(243.5)	719.8	(292.0)
Journey Escape	-5,592.9*	(1,253.2)*	-5,121.6*	(5,952.6)*	-4,654.0*	(5,446.3)*	-2,338.9*	(952.8)*
Kangaroo	1,305.6	(555.7)	2,442.5	(1,282.4)	3,152.8	(1,546.3)	4,225.8	(2,046.9)
Krull	4,922.1	(1,703.7)	6,762.2	(5,168.9)	7,491.5	(5,823.9)	8,894.9	(8,482.7)
Kung-Fu Master	20,679.3	(2,246.8)	23,548.3	(2,926.8)	26,745.6	(3,281.5)	29,915.5	(3,647.5)
Mont. Revenge	117.1	(175.3)	520.8	(486.7)	567.7	(588.9)	574.2	(590.1)
Ms. Pac-Man	2,626.7	(521.1)	3,446.0	(462.9)	3,916.6	(542.5)	4,440.5	(616.4)
Name This Game	4,626.2	(284.3)	6,164.2	(357.0)	6,219.8	(1,821.2)	6,750.3	(1,376.5)
Phoenix	2,319.4	(534.0)	4,579.9	(303.9)	4,247.3	(1,360.1)	5,197.0	(374.5)
Pitfall!	-0.3	(1.2)	-0.1	(0.3)	0.0	(0.0)	0.0	(0.0)
Pong†	1.8	(3.9)	10.9	(3.3)	12.6	(2.8)	14.5	(2.0)
Pooyan	1,347.2	(121.5)	1,820.5	(107.5)	2,006.7	(159.4)	2,197.8	(133.7)
Private Eye	36.7	(46.3)	36.2	(49.2)	27.9	(44.9)	44.2	(49.3)
Q*bert†	3,535.9	(745.2)	4,605.7	(567.3)	5,931.9	(1,174.4)	6,992.9	(1,479.0)
River Raid	4,141.9	(574.4)	7,399.5	(492.5)	8,988.3	(1,154.3)	10,639.2	(1,882.6)
Road Runner	18,258.0	(2,876.9)	23,380.2	(5,940.3)	28,453.4	(3,227.4)	31,493.9	(4,160.7)
Robotank	21.3	(3.4)	25.0	(2.8)	26.3	(2.7)	27.3	(2.3)
Seaquest†	788.2	(225.2)	1,201.6	(178.4)	1,319.2	(356.5)	1,402.9	(328.3)
Skiing	-29,965.4	(59.9)	-29,955.6	(50.7)	-29,955.9	(57.7)	-29,940.2	(102.2)
Solaris	480.9	(185.1)	585.5	(213.3)	704.2	(264.3)	807.3	(216.2)
Space Invaders†	466.2	(30.4)	579.1	(36.5)	656.7	(67.6)	759.4	(58.7)
Star Gunner	1,002.1	(64.6)	1,014.1	(72.7)	1,058.1	(101.0)	1,107.6	(125.6)
Tennis	-0.1	(0.1)	-0.1	(0.0)	-0.1	(0.0)	-0.1	(0.0)
Time Pilot	3,439.5	(503.4)	3,997.8	(436.4)	4,112.0	(289.4)	4,221.5	(402.1)
Tutankham	122.2	(3.5)	152.7	(16.0)	88.9	(64.1)	91.5	(63.3)
Up and Down	10,580.2	(3,446.4)	10,049.1	(9,340.7)	11,514.5	(11,988.8)	15,400.1	(14,864.6)
Venture	0.0	(0.0)	0.0	(0.0)	53.8	(263.7)	139.3	(323.2)
Video Pinball	11,271.1	(1,142.7)	13,259.3	(1,327.2)	14,334.7	(1,097.4)	13,398.0	(3,643.7)
Wizard of Wor	1,975.9	(471.4)	2,738.8	(613.3)	3,247.5	(713.0)	2,043.5	(801.3)
Yar's Revenge	4,961.2	(1,200.2)	5,460.2	(1,145.0)	6,073.9	(1,052.9)	7,257.8	(1,884.8)
Zaxxon	1,180.9	(618.8)	4,539.6	(1,401.0)	6,701.4	(1,974.3)	8,166.8	(3,979.8)

Table 8: Sarsa(λ) + Blob-PROST results across 60 games. See Appendix B for details.

554

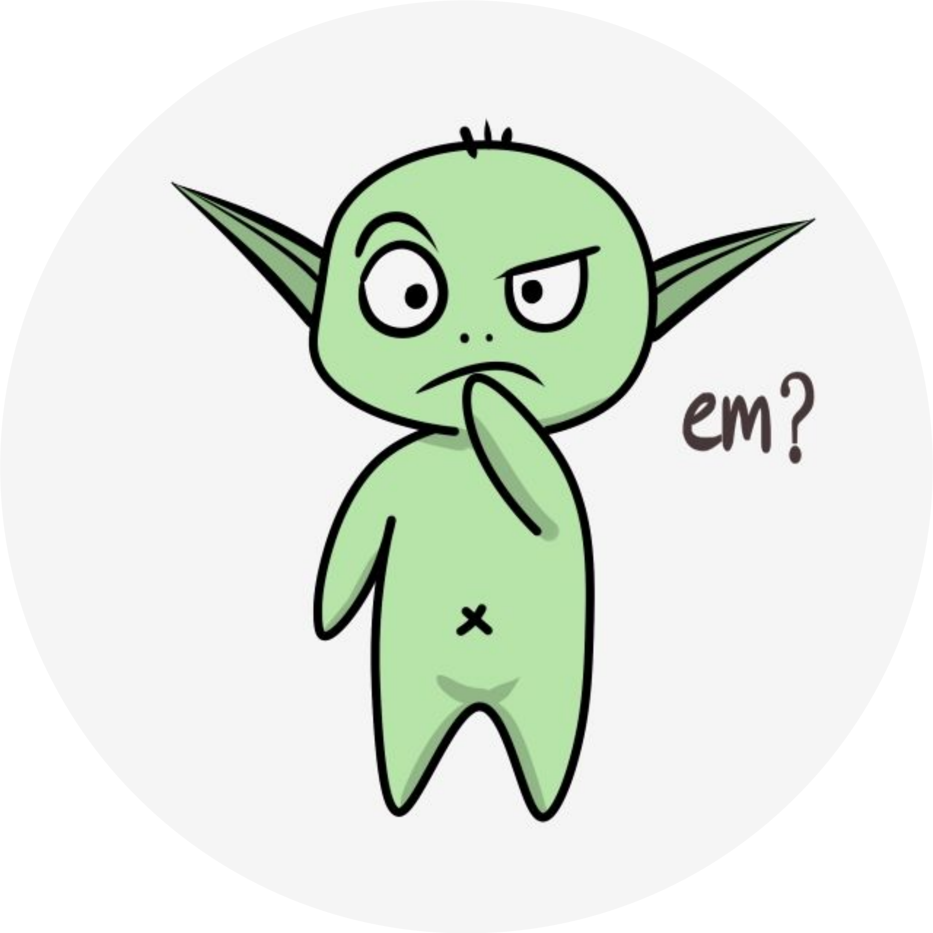


Revisiting the ALE: Evaluation Protocols and Open Problems

Game	10M frames		50M frames		100M frames		200M frames	
Alien	600.5	(23.6)	1,426.6	(81.6)	1,952.6	(216.0)	2,742.0	(357.5)
Amidar	91.6	(10.5)	414.2	(53.6)	621.6	(92.6)	792.6	(220.4)
Assault	688.9	(16.0)	1,327.5	(83.9)	1,433.9	(126.6)	1,424.6	(106.8)
Asterix†	1,732.6	(314.6)	3,122.6	(96.4)	3,423.4	(213.6)	2,866.8	(1,354.6)
Asteroids	301.4	(14.3)	458.1	(28.5)	458.0	(18.9)	528.5	(37.0)
Atlantis	6,639.4	(208.4)	51,324.4	(8,681.7)	291,134.7	(31,575.2)	232,442.9	(128,678.4)
Bank Heist	32.3	(6.5)	448.2	(104.8)	740.7	(130.6)	760.0	(82.3)
Battle Zone	2,428.3	(200.4)	10,838.4	(1,807.6)	15,048.5	(2,372.0)	20,547.5	(1,843.0)
Beam Rider †	693.9	(111.0)	4,551.5	(849.1)	4,977.2	(292.2)	5,700.5	(362.5)
Berzerk	434.5	(51.2)	457.5	(9.4)	470.0	(24.5)	487.2	(29.9)
Bowling	28.7	(0.8)	29.4	(1.8)	32.8	(3.6)	33.6	(2.7)
Boxing	18.6	(3.8)	71.7	(2.7)	77.9	(0.5)	72.7	(4.9)
Breakout	14.2	(1.2)	75.1	(4.3)	57.9	(14.6)	35.1	(22.6)
Carnival	588.5	(47.0)	2,131.6	(534.3)	4,621.9	(191.0)	4,803.8	(189.0)
Centipede	3,075.2	(381.1)	2,280.0	(184.2)	2,555.2	(195.1)	2,838.9	(225.3)
Chopper Comm.	841.4	(144.3)	2,104.8	(327.7)	3,288.1	(339.2)	4,399.6	(401.5)
Crazy Climber	43,716.6	(2,571.2)	80,599.6	(4,209.8)	64,807.3	(26,100.0)	78,352.1	(1,967.3)
Defender	2,409.9	(78.6)	2,525.7	(124.0)	2,711.6	(96.8)	2,941.3	(106.2)
Demon Attack	154.8	(11.5)	3,744.6	(688.9)	4,556.5	(947.2)	5,182.0	(778.0)
Double Dunk	-20.9	(0.3)	-18.4	(1.2)	-15.6	(1.6)	-8.7	(4.5)
Elevator Action	6.7	(13.3)	4.5	(9.0)	4.7	(9.4)	6.0	(10.4)
Enduro	473.2	(22.3)	578.0	(79.6)	597.4	(153.1)	688.2	(32.4)
Fishing Derby	-63.1	(7.8)	7.5	(4.1)	12.2	(1.4)	10.2	(1.9)
Freeway†	13.8	(8.1)	31.7	(0.7)	32.4	(0.3)	33.0	(0.3)
Frostbite	241.8	(30.8)	292.5	(28.8)	274.3	(8.8)	279.6	(13.9)
Gopher	679.6	(35.2)	2,233.7	(123.1)	2,988.8	(514.4)	3,925.5	(521.4)
Gravitar	79.5	(8.0)	109.3	(3.1)	118.5	(22.0)	154.9	(17.7)
H.E.R.O.	1,667.9	(1,107.8)	11,564.0	(3,722.4)	14,684.7	(1,840.6)	18,843.3	(2,234.9)
Ice Hockey	-15.1	(0.3)	-8.9	(1.7)	-4.4	(2.0)	-3.8	(4.7)
James Bond	30.7	(6.0)	191.4	(144.9)	517.2	(35.8)	581.0	(21.3)
Journey Escape	-2,220.0	(176.1)	-2,409.7	(341.2)	-2,959.0	(383.9)	-3,503.0	(488.5)
Kangaroo	298.6	(56.1)	8,878.8	(2,886.1)	12,846.9	(688.3)	12,291.7	(1,115.9)
Krull	4,424.7	(492.7)	6,035.6	(248.6)	6,589.8	(264.4)	6,416.0	(128.5)
Kung-Fu Master	9,468.1	(1,975.9)	17,537.4	(1,128.8)	17,772.3	(3,423.3)	16,472.7	(2,892.7)
Mont. Revenge	0.2	(0.4)	0.2	(0.4)	0.0	(0.0)	0.0	(0.0)
Ms. Pac-Man	1,675.5	(41.9)	2,626.1	(139.8)	2,964.9	(100.8)	3,116.2	(141.2)
Name This Game	2,265.6	(171.0)	4,105.4	(932.3)	4,105.6	(653.5)	3,925.2	(660.2)
Phoenix	1,501.2	(278.1)	3,174.0	(543.5)	2,607.1	(644.1)	2,831.0	(581.0)
Pitfall!	-24.9	(14.8)	-28.2	(13.0)	-23.3	(9.6)	-21.4	(3.2)
Pong	-15.9	(1.0)	12.2	(1.0)	15.2	(0.7)	15.1	(1.0)
Pooyan	2,278.9	(273.7)	3,528.9	(256.3)	3,387.8	(182.8)	3,700.4	(349.5)
Private Eye	81.6	(15.6)	60.4	(92.4)	1,447.4	(2,567.9)	3,967.5	(5,540.6)
Q*bert	674.7	(53.6)	3,142.1	(1,238.7)	7,585.4	(2,787.4)	9,875.5	(1,385.3)
River Raid	3,166.2	(125.2)	8,738.1	(500.0)	10,733.1	(229.9)	10,210.4	(435.0)
Road Runner	14,742.2	(1,553.4)	37,271.7	(1,234.5)	41,918.4	(1,762.5)	42,028.3	(1,492.0)
Robotank	4.1	(0.3)	28.4	(1.4)	38.0	(1.6)	58.0	(6.4)
Seaquest†	311.5	(36.9)	1,430.8	(162.3)	1,573.4	(561.4)	1,485.7	(740.8)
Skiing	-20,837.5	(1,550.2)	-17,545.5	(4,041.5)	-13,365.1	(800.7)	-12,446.6	(1,257.9)
Solaris	1,030.2	(40.3)	977.7	(112.5)	783.4	(55.3)	1,210.0	(148.3)
Space Invaders†	211.6	(14.8)	686.6	(37.0)	787.2	(173.3)	823.6	(335.0)
Star Gunner	603.0	(28.0)	1,492.3	(79.7)	11,590.5	(4,658.9)	39,269.9	(5,298.8)
Tennis	-23.8	(0.1)	-23.9	(0.1)	-23.9	(0.0)	-23.9	(0.0)
Time Pilot	1,078.8	(60.3)	1,068.1	(138.8)	1,330.7	(177.1)	2,061.8	(228.8)
Tutankham	56.5	(10.0)	64.9	(12.6)	65.1	(11.9)	60.0	(12.7)
Up and Down	4,378.4	(172.5)	6,718.3	(671.2)	5,962.8	(618.7)	4,750.7	(1,007.5)
Venture	24.4	(46.9)	21.4	(15.1)	4.4	(5.4)	3.2	(4.7)
Video Pinball	4,009.3	(271.9)	7,817.0	(1,884.4)	16,626.2	(3,740.6)	15,398.5	(2,126.1)
Wizard of Wor	184.2	(22.0)	1,377.4	(71.0)	1,440.6	(237.3)	2,231.1	(820.8)
Yar's Revenge	7,261.4	(777.1)	10,344.8	(452.4)	10,312.3	(528.9)	13,073.4	(1,961.8)
Zaxxon	53.5	(51.0)	672.3	(748.5)	1,638.2	(784.0)	3,852.1	(1,120.7)

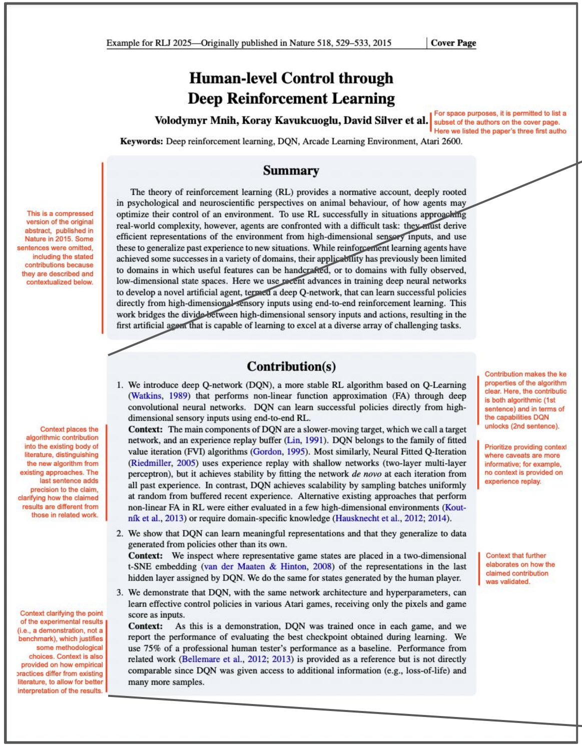
Table 9: DQN results across 60 games. See Appendix B for details.

555



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

We can always update our beliefs



Example for RLJ 2025—Originally published in Nature 518, 529–533, 2015 | **Cover Page**

Human-level Control through Deep Reinforcement Learning

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al.

For space purposes, it is permitted to list a subset of the authors on the cover page. Here we listed the paper's three first autho

Keywords: Deep reinforcement learning, DQN, Arcade Learning Environment, Atari 2600.

This is a compressed version of the original abstract, published in Nature in 2015. Some sentences were omitted, including the stated contributions because they are described and contextualized below.

Summary

The theory of reinforcement learning (RL) provides a normative account, deeply rooted in psychological and neuroscientific perspectives on animal behaviour, of how agents may optimize their control of an environment. To use RL successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations. While reinforcement learning agents have achieved some successes in a variety of domains, their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. Here we use recent advances in training deep neural networks to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. This work bridges the divide between high-dimensional sensory inputs and actions, resulting in the first artificial agent that is capable of learning to excel at a diverse array of challenging tasks.

Contribution(s)

1. We introduce deep Q-network (DQN), a more stable RL algorithm based on Q-Learning (Watkins, 1989) that performs non-linear function approximation (FA) through deep convolutional neural networks. DQN can learn successful policies directly from high-dimensional sensory inputs using end-to-end RL.
   Context: The main components of DQN are a slower-moving target, which we call a target network, and an experience replay buffer (Lin, 1991). DQN belongs to the family of fitted value iteration (FVI) algorithms (Gordon, 1995). Most similarly, Neural Fitted Q-Iteration (Riedmiller, 2005) uses experience replay with shallow networks (two-layer multi-layer perceptron), but it achieves stability by fitting the network *de novo* at each iteration from all past experience. In contrast, DQN achieves scalability by sampling batches uniformly at random from buffered recent experience. Alternative existing approaches that perform non-linear FA in RL were either evaluated in a few high-dimensional environments (Koutník et al., 2013) or require domain-specific knowledge (Hausknecht et al., 2012; 2014).
2. We show that DQN can learn meaningful representations and that they generalize to data generated from policies other than its own.
   Context: We inspect where representative game states are placed in a two-dimensional t-SNE embedding (van der Maaten & Hinton, 2008) of the representations in the last hidden layer assigned by DQN. We do the same for states generated by the human player.
3. We demonstrate that DQN, with the same network architecture and hyperparameters, can learn effective control policies in various Atari games, receiving only the pixels and game score as inputs.
   Context: As this is a demonstration, DQN was trained once in each game, and we report the performance of evaluating the best checkpoint obtained during learning. We use 75% of a professional human tester's performance as a baseline. Performance from related work (Bellemare et al., 2012; 2013) is provided as a reference but is not directly comparable since DQN was given access to additional information (e.g., loss-of-life) and many more samples.

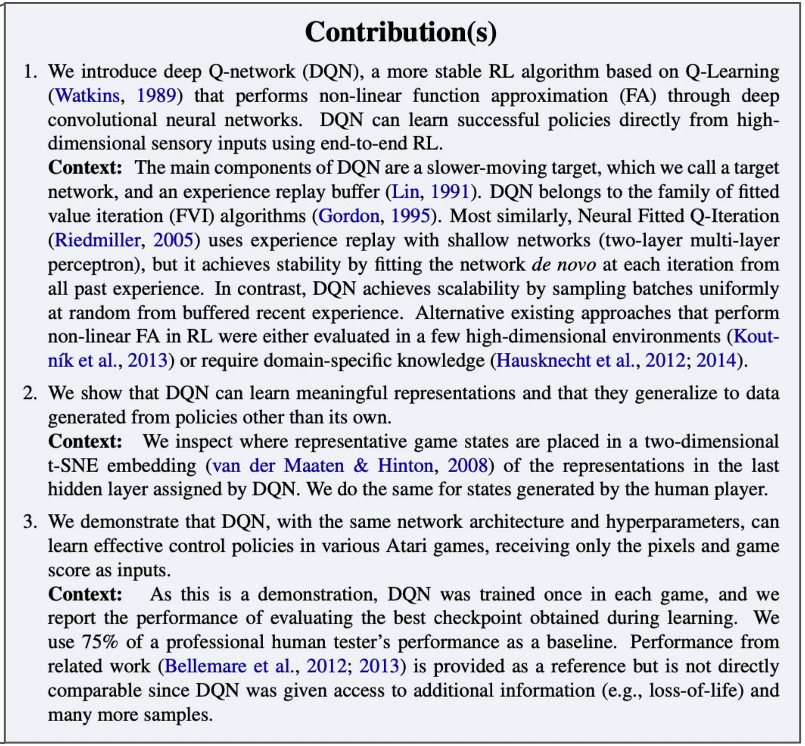
Contribution makes the ke properties of the algorithm clear. Here, the contributic is both algorithmic (1st sentence) and in terms of the capabilities DQN unlocks (2nd sentence).

Context places the algorithmic contribution into the existing body of literature, distinguishing the new algorithm from existing approaches. The last sentence adds precision to the claim, clarifying how the claimed results are different from those in related work.

Prioritize providing contex where caveats are more informative; for example, no context is provided on experience replay.

Context that further elaborates on how the claimed contribution was validated.

Context clarifying the point of the experimental results (i.e., a demonstration, not a benchmark), which justifies some methodological choices. Context is also provided on how empirical ractices differ from existing literature, to allow for better interpretation of the results.



Contribution(s)

1. We introduce deep Q-network (DQN), a more stable RL algorithm based on Q-Learning (Watkins, 1989) that performs non-linear function approximation (FA) through deep convolutional neural networks. DQN can learn successful policies directly from high-dimensional sensory inputs using end-to-end RL.
   Context: The main components of DQN are a slower-moving target, which we call a target network, and an experience replay buffer (Lin, 1991). DQN belongs to the family of fitted value iteration (FVI) algorithms (Gordon, 1995). Most similarly, Neural Fitted Q-Iteration (Riedmiller, 2005) uses experience replay with shallow networks (two-layer multi-layer perceptron), but it achieves stability by fitting the network *de novo* at each iteration from all past experience. In contrast, DQN achieves scalability by sampling batches uniformly at random from buffered recent experience. Alternative existing approaches that perform non-linear FA in RL were either evaluated in a few high-dimensional environments (Koutník et al., 2013) or require domain-specific knowledge (Hausknecht et al., 2012; 2014).
2. We show that DQN can learn meaningful representations and that they generalize to data generated from policies other than its own.
   Context: We inspect where representative game states are placed in a two-dimensional t-SNE embedding (van der Maaten & Hinton, 2008) of the representations in the last hidden layer assigned by DQN. We do the same for states generated by the human player.
3. We demonstrate that DQN, with the same network architecture and hyperparameters, can learn effective control policies in various Atari games, receiving only the pixels and game score as inputs.
   Context: As this is a demonstration, DQN was trained once in each game, and we report the performance of evaluating the best checkpoint obtained during learning. We use 75% of a professional human tester's performance as a baseline. Performance from related work (Bellemare et al., 2012; 2013) is provided as a reference but is not directly comparable since DQN was given access to additional information (e.g., loss-of-life) and many more samples.

Next class

- What I plan to do:
  - Value-based Model-free Methods Objective Functions: Double Learning

- What I recommend YOU to do for next class:
  - Read the lecture notes and the Double DQN paper.

    van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double Q-learning. In Proceedings of the Conference on Artificial Intelligence, pages 2094–2100. Preprint made available on September 22, 2015.
  - Start thinking about Assignment 2!