When to use parametric models in reinforcement learning?

Hado van Hasselt, DeepMind

Tea-time talk, Montreal, July 11, 2019

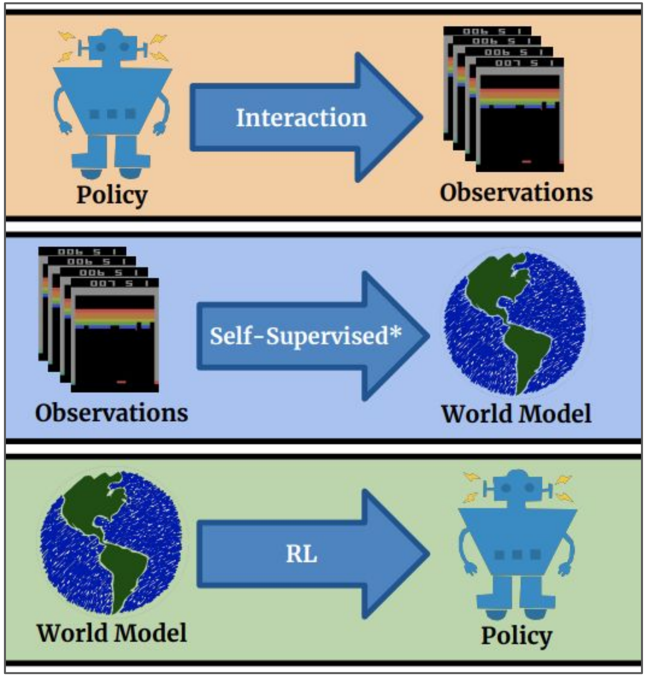
Related paper: https://arxiv.org/abs/1906.05243

Motivation

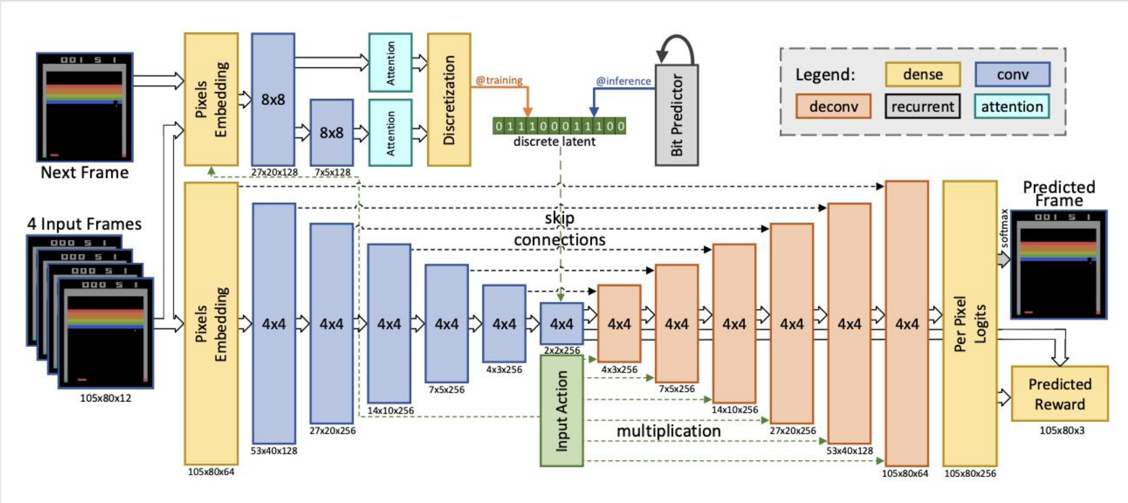
Model Based Reinforcement Learning for Atari

Łukasz Kaiser [*1] **Mohammad Babaeizadeh** [*2 3] **Piotr Miłoś** [*4 5] **Błażej Osiński** [*4 5 3]
Roy H Campbell [2] **Konrad Czechowski** [4] **Dumitru Erhan** [1] **Chelsea Finn** [1] **Piotr Kozakowski** [4] **Sergey Levine** [1]
Afroz Mohiuddin [1] **Ryan Sepassi** [1] **George Tucker** [1] **Henryk Michalewski** [4 5]

Motivation



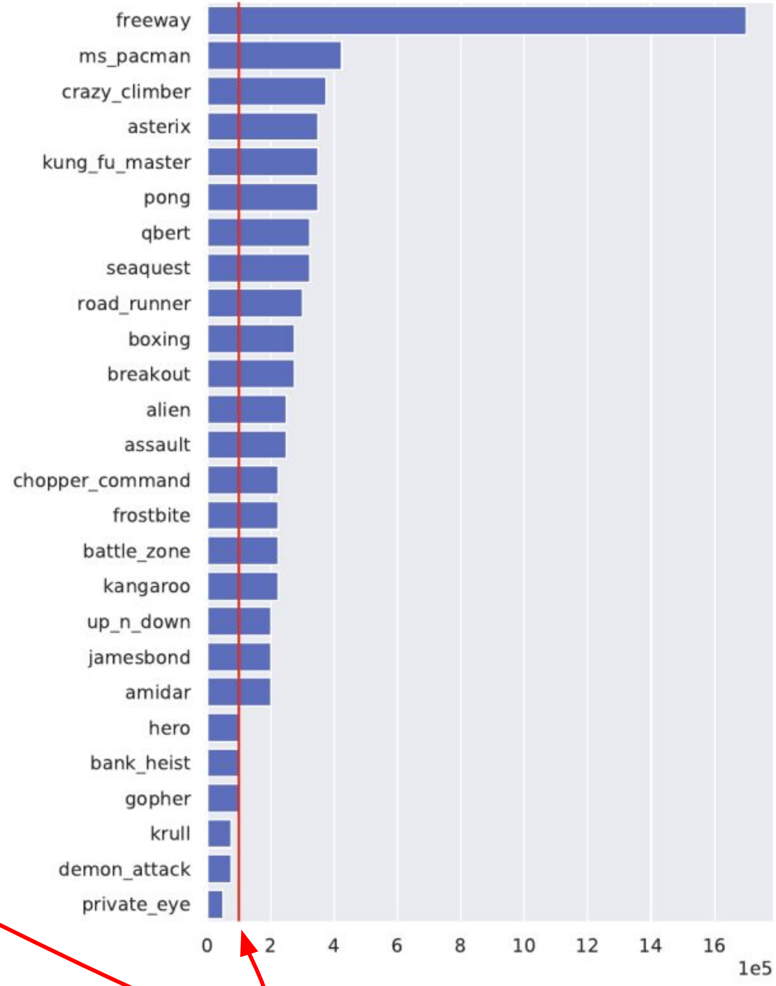
Policy
Interaction
Observations
Observations
Self-Supervised*
World Model
World Model
RL
Policy



Next Frame
4 Input Frames
105x80x12
Pixels Embedding
8x8
8x8
27x20x128
7x5x128
Attention
Attention
Discretization
@training
@inference
0 1 1 1 0 0 0 1 1 1 0 0
discrete latent
Bit Predictor
Legend:
dense
conv
deconv
recurrent
attention
skip connections
multiplication
Pixels Embedding
105x80x64
4x4
53x40x128
27x20x256
14x10x256
7x5x256
4x3x256
2x2x256
4x3x256
7x5x256
14x10x256
27x20x256
53x40x128
105x80x64
Input Action
Per Pixel Logits
105x80x256
softmax
Predicted Frame
Predicted Reward
105x80x3

Motivation

Performance (=score)
after 100,000 steps (=400,000 frames)



Baseline: Rainbow DQN

Question

Why does the parametric model perform better than replay?

Models and planning

A **model** is a function:

$$r,\ s' = m(s, a)$$

We can use models to **plan**: spend more compute to improve prediction & policies.

We can also plan with **experience replay**:

$$r_{n+1},\ s_{n+1} = \text{replay}(s_n, a_n)$$

- Experience replay is similar to a non-parametric model
- But we can only query it at observed state action pairs (s_n, a_n), $n < t$.

Replay and models, properties

Typically models use **less memory** and **more compute** than replay

But what about **data efficiency & performance?**

Algorithms

for iteration $\in \{1, 2, \ldots, K\}$ **do**
  for interaction $\in \{1, 2, \ldots, M\}$ **do**
    Generate action: $a \leftarrow \pi(s)$
    Generate reward, next state: $r, s' \leftarrow \mathcal{E}(a)$
    $m, d \leftarrow \textsc{UpdateModel}(s, a, r, s')$
    $\pi, v \leftarrow \textsc{UpdateAgent}(s, a, r, s')$
    Update current state: $s \leftarrow s'$
  end for
  for planning step $\in \{1, 2, \ldots, P\}$ **do**
    Generate state, action $\tilde{s}, \tilde{a} \leftarrow d$
    Generate reward, next state: $\tilde{r}, \tilde{s}' \leftarrow m(\tilde{s}, \tilde{a})$
    $\pi, v \leftarrow \textsc{UpdateAgent}(\tilde{s}, \tilde{a}, \tilde{r}, \tilde{s}')$
  end for

K: iterations
M: real steps / iteration
P: planned steps / iteration

State-sampling distribution

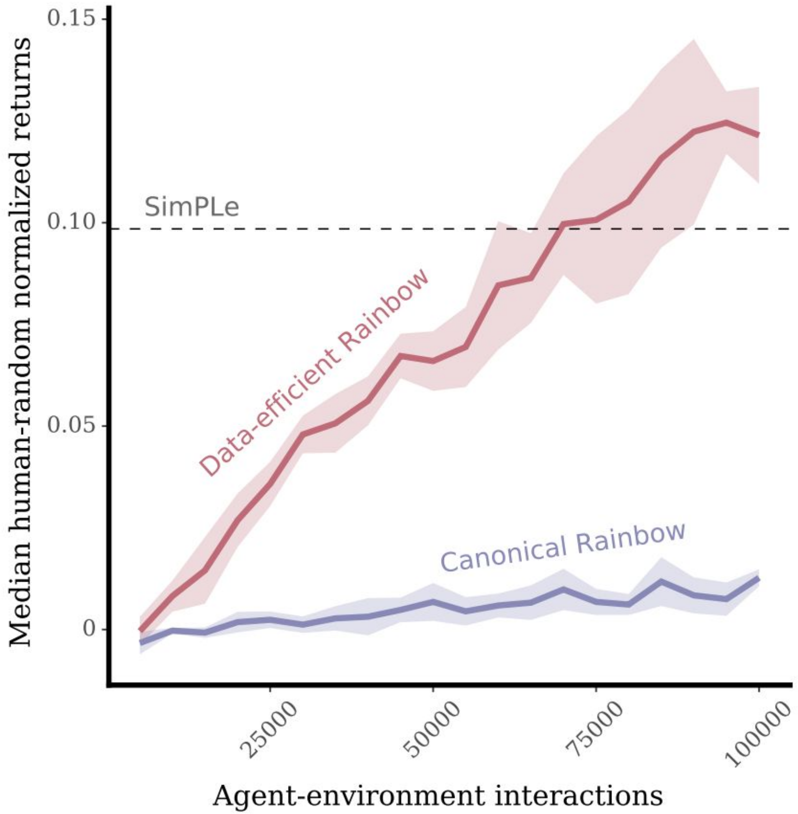
Algorithms

	Iterations (K)	Real steps per iteration (M)	Planned steps per iteration (P)
SimPLe	16	6400	800,000
Rainbow DQN	12,500,000	4	32
Data-efficient Rainbow DQN	100,000	1	32

Algorithms

	Total real experience (**K** x **M**)	Total planned experience (**K** x **P**)
SimPLe	**100,000** (400K frames)	**15,200,000**
Rainbow DQN	**50,000,000** (200M frames)	**400,000,000**
Data-efficient Rainbow DQN	**100,000** (400K frames)	**3,200,000**

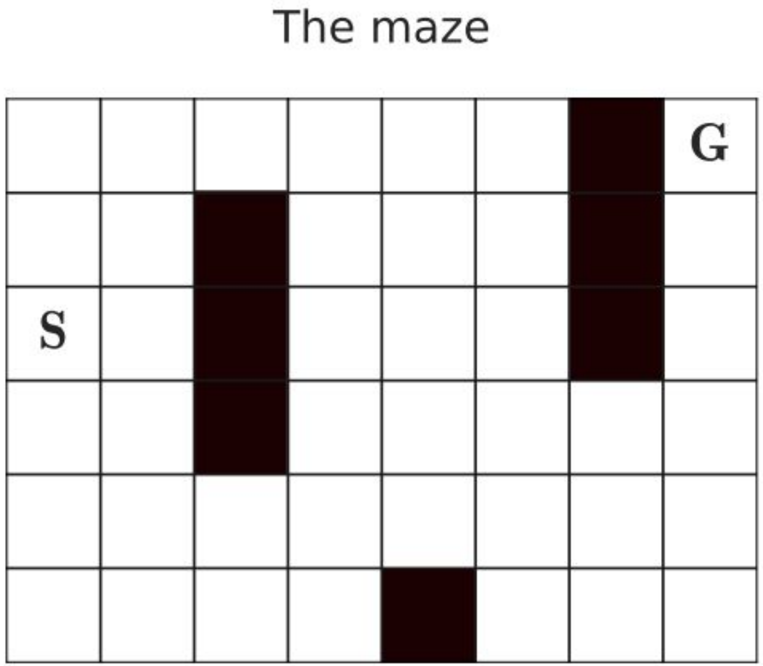
Algorithms

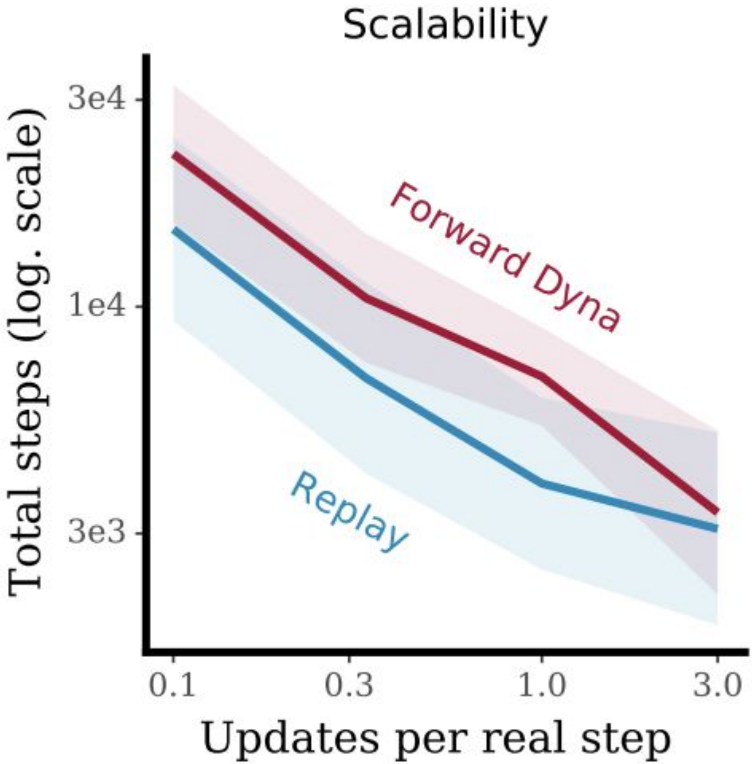


0.15
0.10
0.05
0
SimPLe
Data-efficient Rainbow
Canonical Rainbow
Median human-random normalized returns
25000
50000
75000
100000
Agent-environment interactions

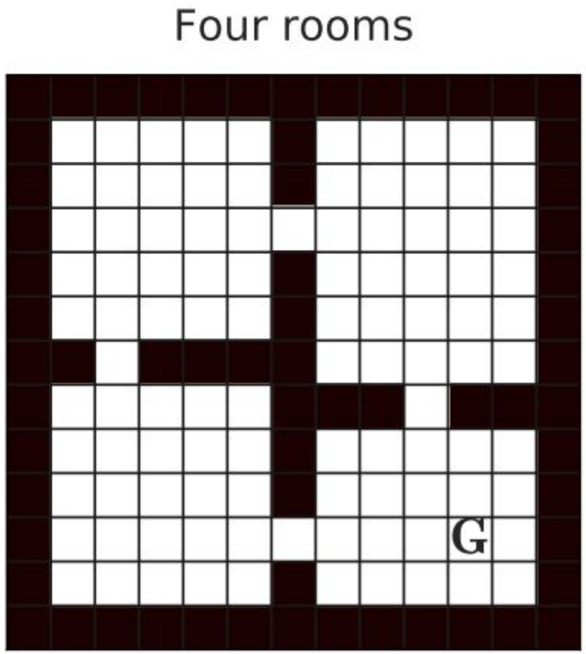
When do models help performance?

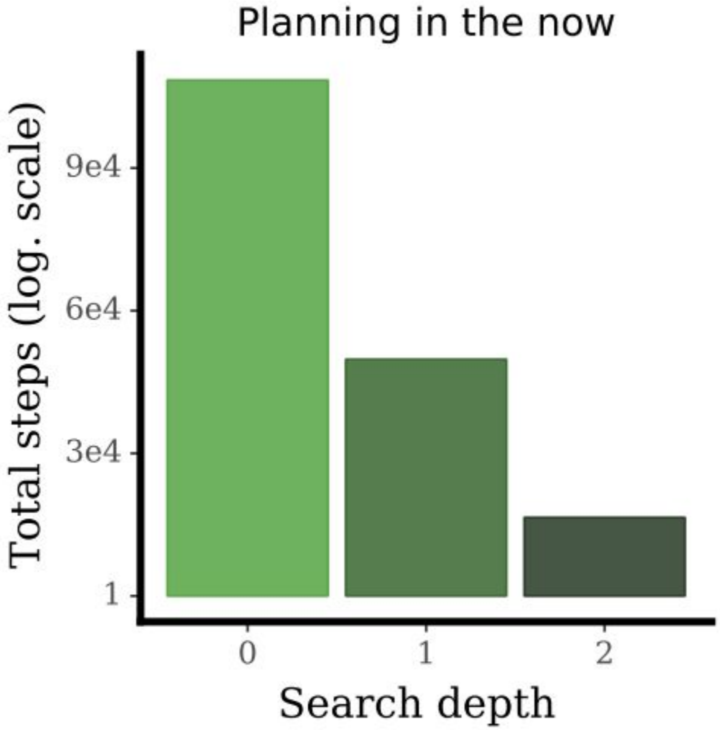
Forward planning for credit assignment



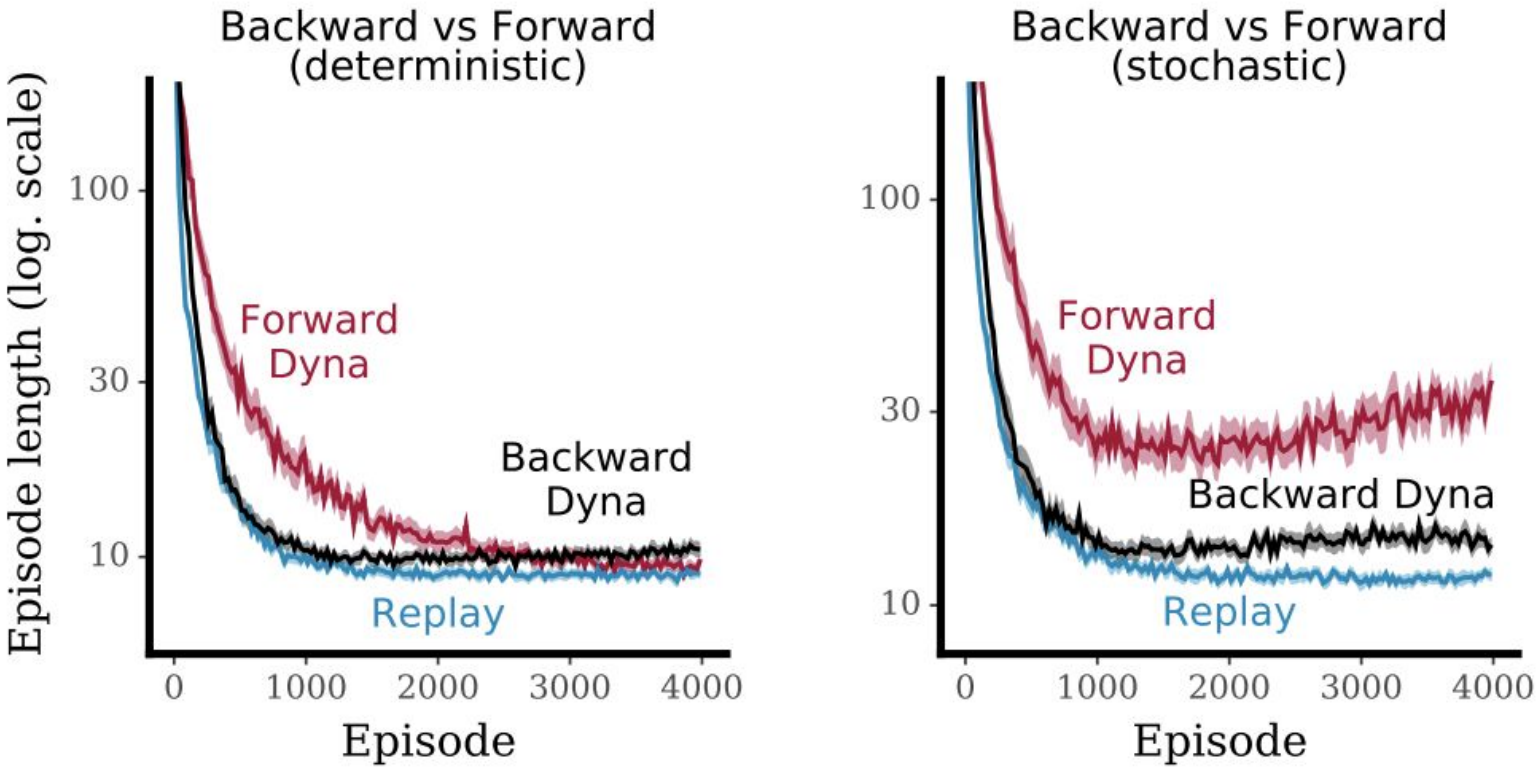


Forward planning for behaviour





Backward planning for credit assignment



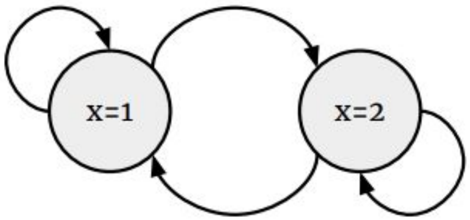
Conclusions

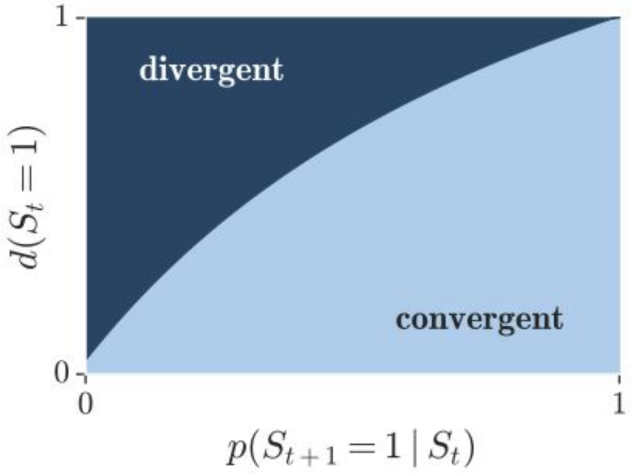
1. **Replay** can be used for **planning**
2. There are different ways to use models:
    a. **Forward** planning for **credit assignment**
    b. **Forward** planning for **immediate behaviour**
    c. **Backward** planning for **credit assignment**

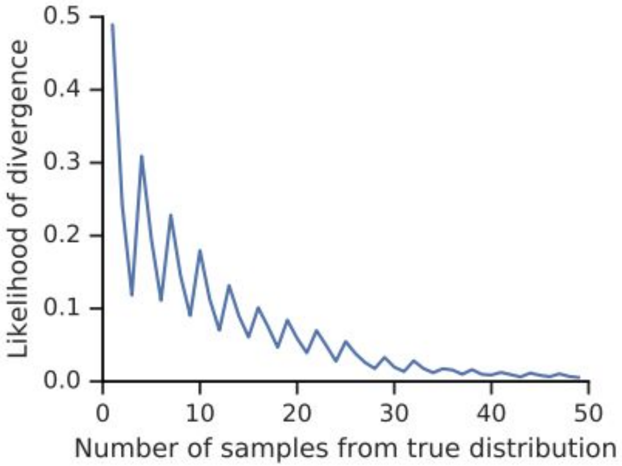
Thank you

More details: https://arxiv.org/abs/1906.05243

Bonus slide: Surprising instabilities







- Even with **perfect models** learning can be unstable
- This happens surprisingly easily!
- Related to the *deadly triad*:
  - the state sampling distribution d and the model m may mismatch, even if $m = p$ is perfect.

Bonus slide: Rainbow DQN hyperparameters changes

Hyper-parameter	canonical	data-efficient
Training frames	200,000,000	400,000
Min replay size for sampling	20,000	1600
Memory size	1,000,000 steps	unbounded
Replay period every	4 steps	1 steps
Multi-step return length	3	20
Q network: channels	32, 64, 64	32, 64
Q network: filter size	$8 \times 8, 4 \times 4, 3 \times 3$	$5 \times 5, 5 \times 5$
Q network: stride	4, 2, 1	5, 5
Q network: hidden units	512	256
Optimizer: learning rate	0.0000625	0.0001