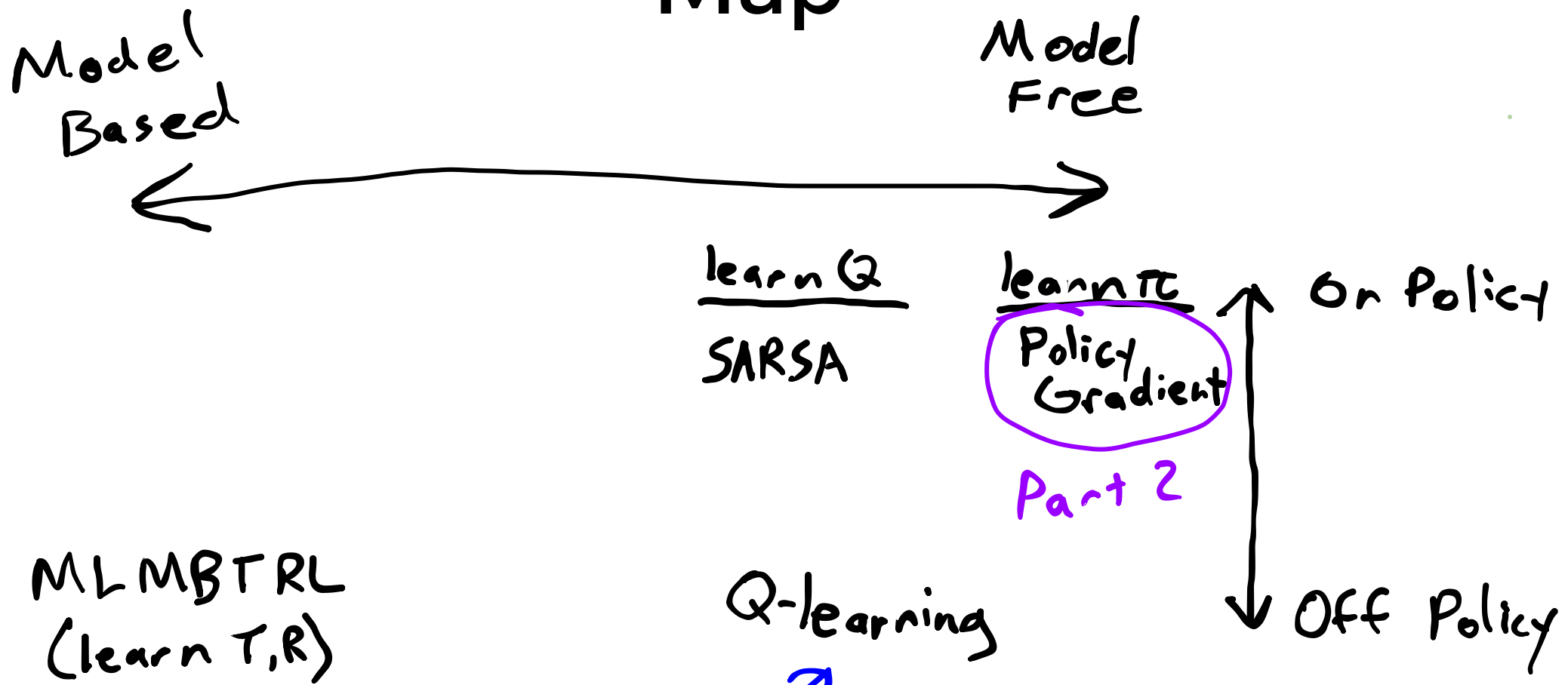
# DQN and Advanced Policy Gradient

# Map

# Map




Challenges:

1. Exploration vs Exploitation
2. Credit Assignment
3. Generalization

Last Time: Neural Networks

# Part I
## DQN

# Q-Learning with Neural Networks

Q-Learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\ (r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Neural Networks

$$\theta^* = \arg\min_\theta \sum_{(x,y)\in\mathcal{D}} l(f_\theta(x), y)$$

$$\theta \leftarrow \theta - \alpha \nabla_\theta\, l(f_\theta(x), y)$$

Deep Q learning:

- Approximate $Q$ with $Q_\theta$
- What should $(x, y)$ be?
- What should $l$ be?

Candidate Algorithm:

loop

$a \leftarrow \text{argmax}\, Q(s,a)$ w.p. $1-\epsilon$, $\quad \text{rand}(A)$ o.w.

$r \leftarrow \text{act!}(\text{env}, a)$

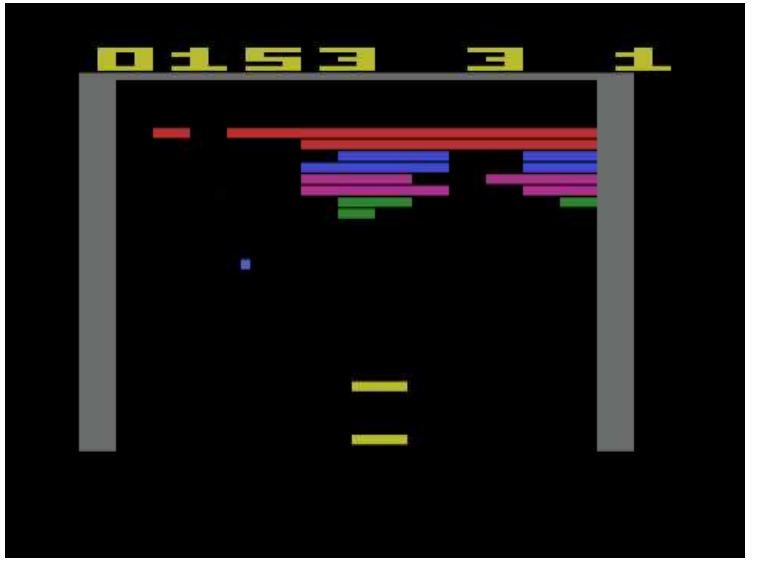
$s' \leftarrow \text{observe}(\text{env})$

$\theta \leftarrow \theta - \alpha \nabla_\theta \left(r + \gamma \max_{a'} Q_\theta(s',a') - Q_\theta(s,a)\right)^2$

$s \leftarrow s'$

# DQN: The Atari Benchmark






# DQN: Problems with Naive Approach

Candidate Algorithm:

loop

$$a \leftarrow \text{argmax}\, Q(s,a) \text{ w.p. } 1-\epsilon, \quad \text{rand}(A) \text{ o.w.}$$

$$r \leftarrow \text{act!}(\text{env}, a)$$

$$s' \leftarrow \text{observe}(\text{env})$$

$$\theta \leftarrow \theta - \alpha \nabla_\theta \left(r + \gamma \max_{a'} Q_\theta(s',a') - Q_\theta(s,a)\right)^2$$

$$s \leftarrow s'$$

Problems:

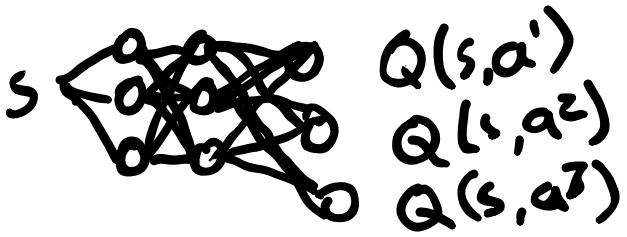
1. Samples Highly Correlated
2. Size-1 batches
3. Moving target

} data buffer / experience replay

} periodically freeze target

# DQN

Q Network Structure:




Experience Tuple: $(s, a, r, s')$

Loss:

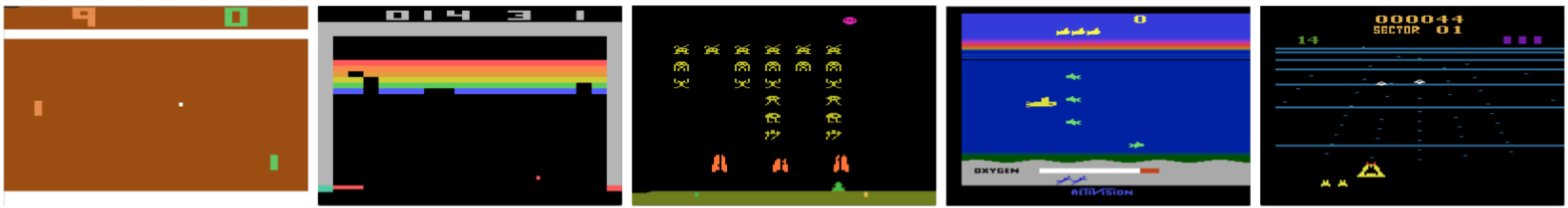
$$l(s, a, r, s') = \left(r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_{\theta}(s, a)\right)^2$$

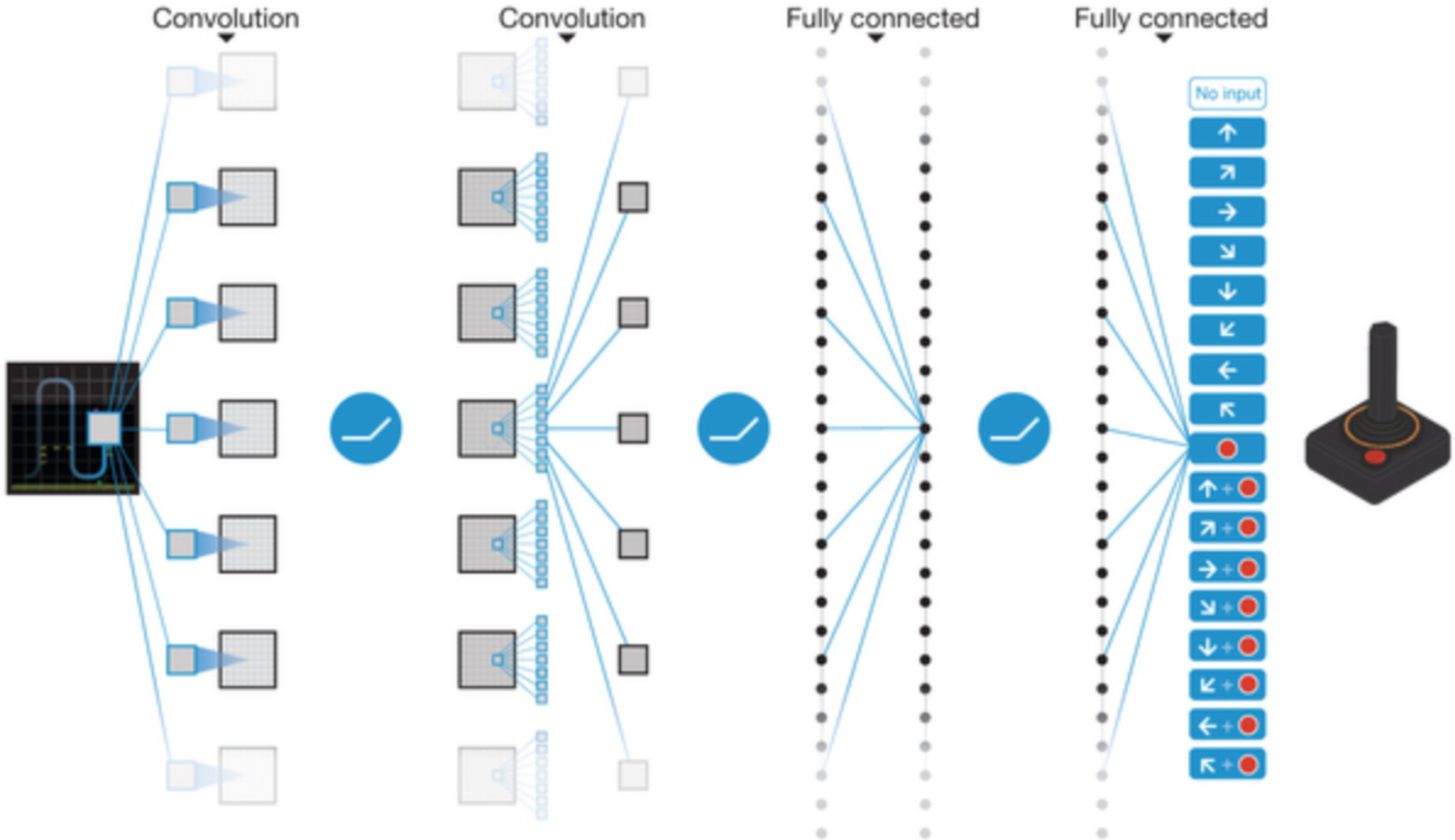
# Playing Atari with Deep Reinforcement Learning

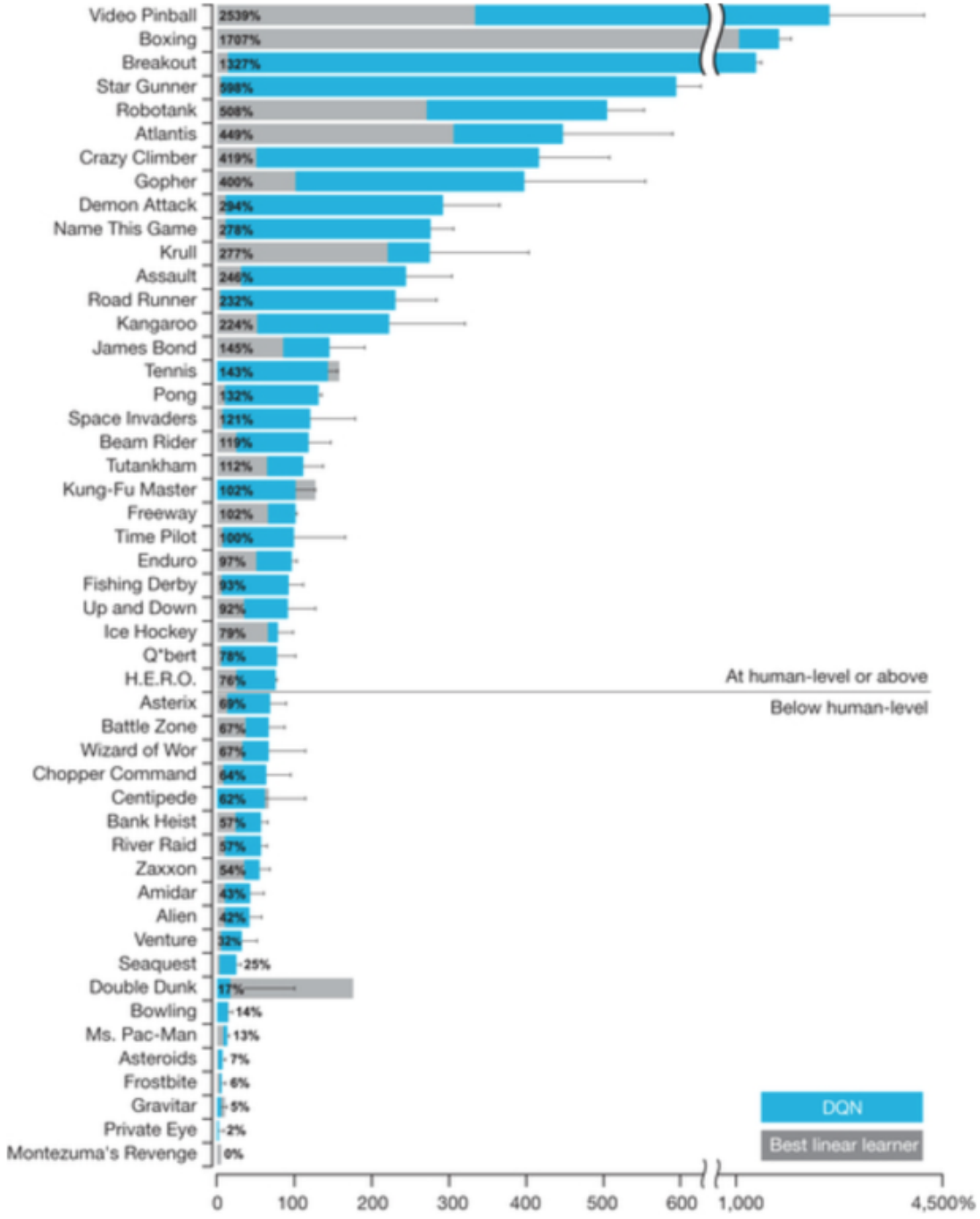
**Volodymyr Mnih** **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

**Daan Wierstra** **Martin Riedmiller**

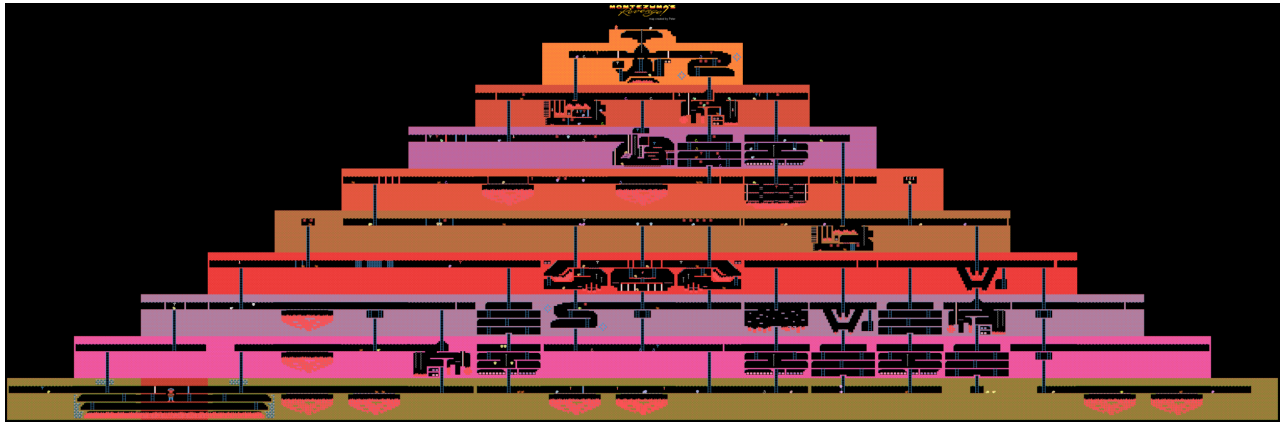
DeepMind Technologies



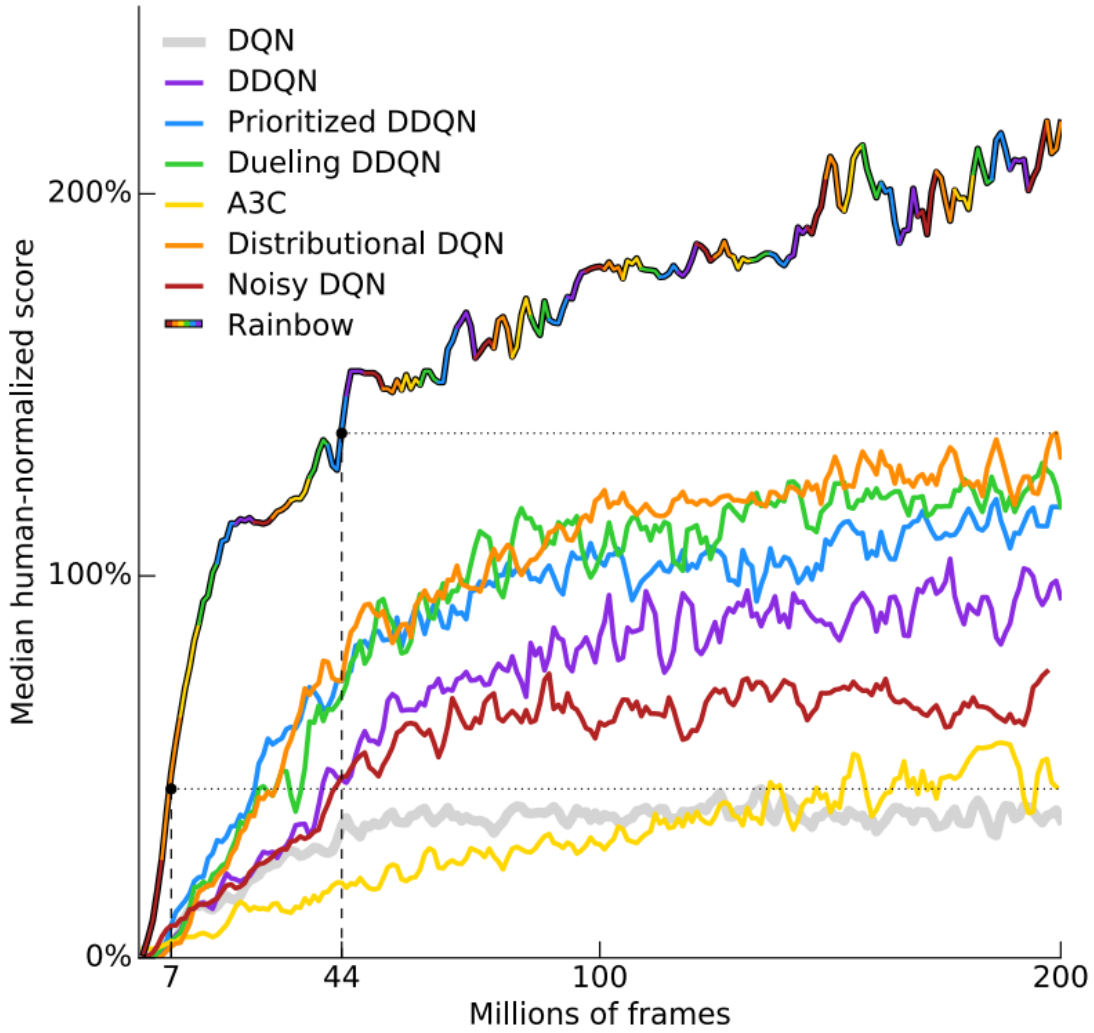

Convolution
Convolution
Fully connected
Fully connected
No input




https://www.youtube.com/watch?v=SuZVyOlgVek



# Rainbow




- Double Q Learning
- Prioritized Replay
  (priority proportional to last TD error)
- Dueling networks
  Value network + advantage network
  $Q(s,a) = V(s) + A(s,a)$
- Multi-step learning
  $(r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma \max Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t))^2$
- Distributional RL
  predict an entire distribution of values instead of just Q
- Noisy Nets

# Actual Learning Curves

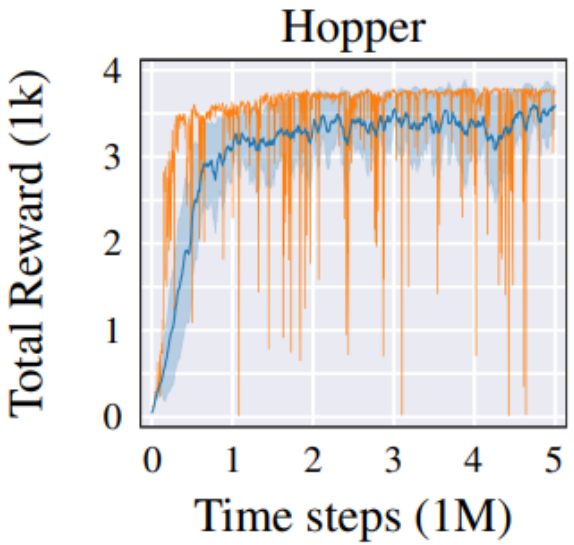



Figure 24: Standard presentation Single seed
The training curve as commonly presented in RL papers and the training curve of a single seed . Both curves are from the TD3 algorithm, trained for 5M time steps. The standard presentation is to evaluate every $N_{\text{freq}}$ steps, average scores over $N_{\text{episodes}}$ evaluation episodes and $N_{\text{seeds}}$ seeds, then to smooth the curve by averaging over a window of $N_{\text{window}}$ evaluations. (In our case this corresponds to $N_{\text{freq}} = 5000$, $N_{\text{episodes}} = 10$, $N_{\text{seeds}} = 10$, $N_{\text{window}} = 10$). The learning curve of a single seed has no smoothing over seeds or evaluations ($N_{\text{seeds}} = 1$, $N_{\text{window}} = 1$). By averaging over many seeds and evaluations, the training curves in RL can appear deceptively smooth and stable.

Paper: For SALE: State-Action Representation Learning for Deep Reinforcement Learning

# Part II
## Improved Policy Gradients

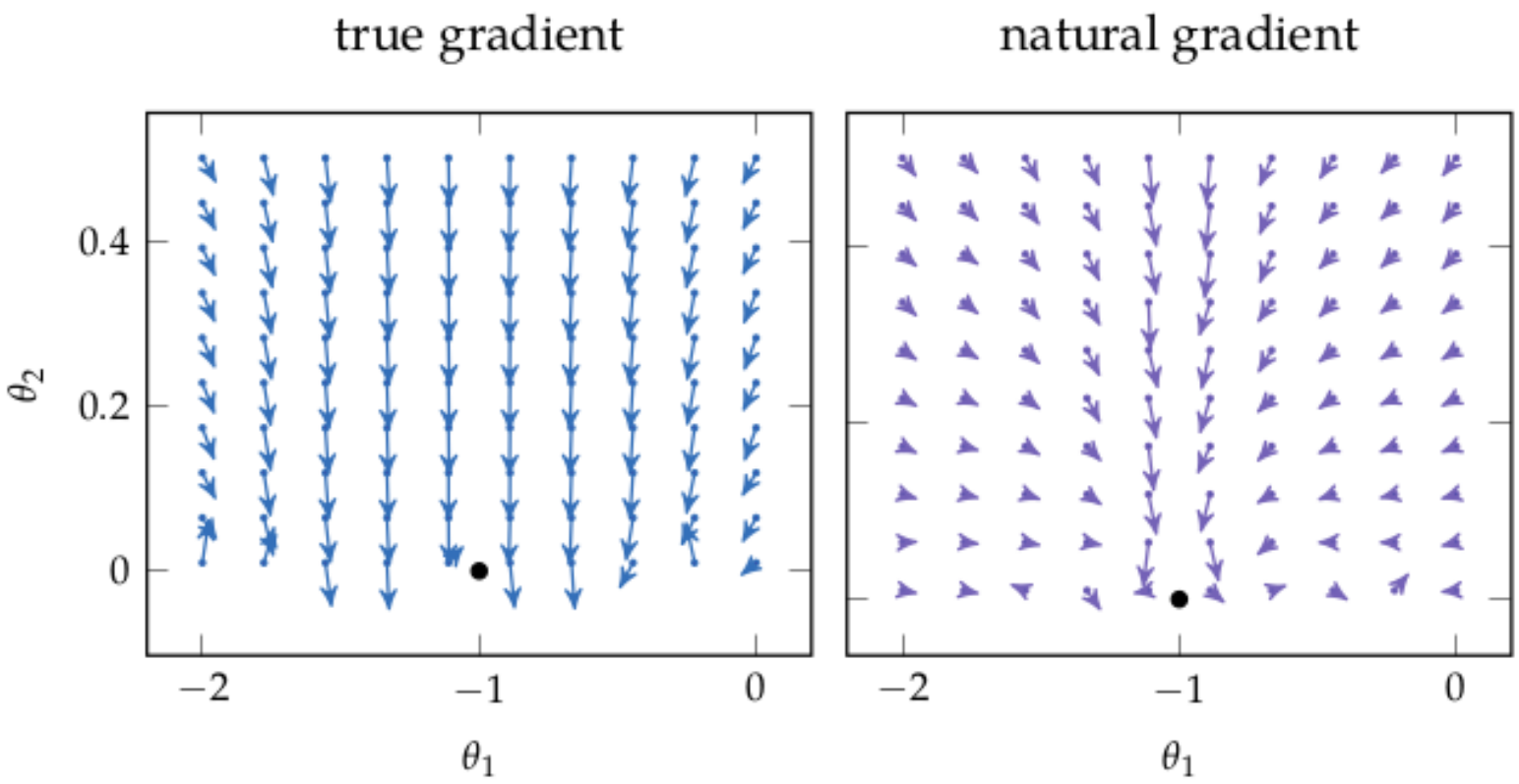
# Restricted Gradient Update

$$\widehat{\nabla U}(\theta) = \sum_{k=0}^{d} \nabla_\theta \log \pi_\theta(a_k \mid s_k) \gamma^k (r_{k,\text{to-go}} - r_{\text{base}}(s_k))$$

$$\theta' = \theta + \alpha \widehat{\nabla U}(\theta)$$

# Natural Gradient

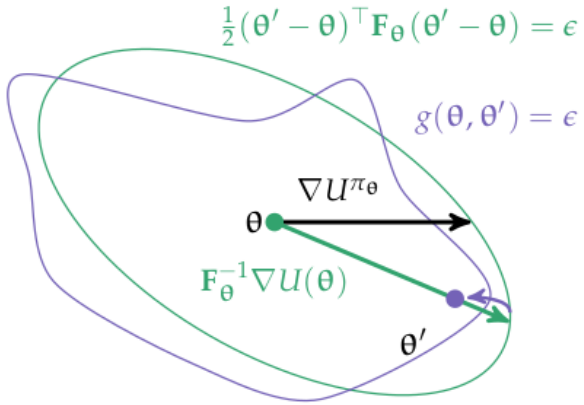
# Natural Gradient

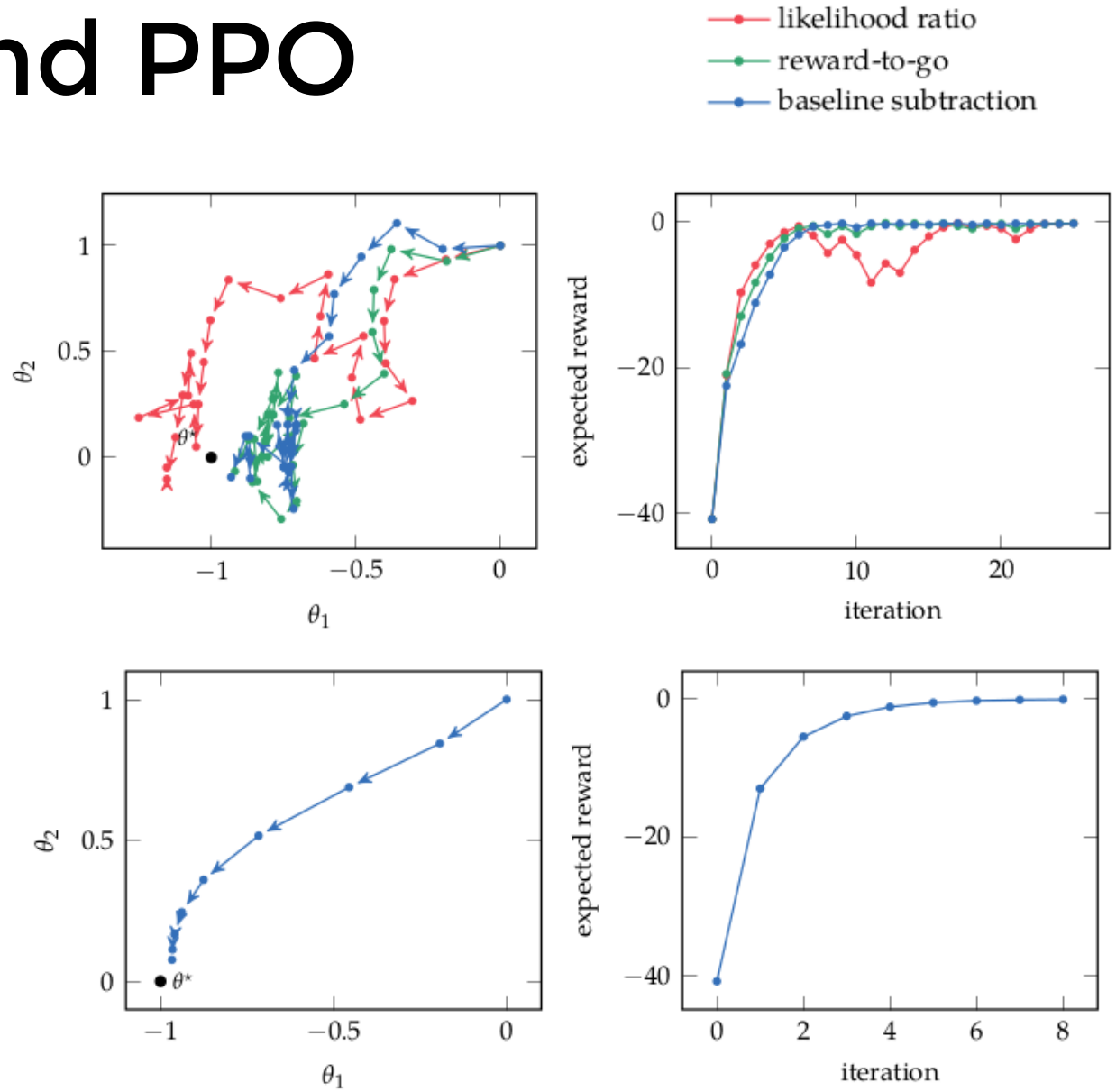


# TRPO and PPO

TRPO = Trust Region Policy Optimization

(Natural gradient + line search)




PPO = Proximal Policy Optimization

(Use clamped surrogate objective to remove the need for line search)

https://openai.com/blog/openai-baselines-ppo/

# Part III
## Actor-Critic

# Actor-Critic

# Actor-Critic

Which should we learn? $A$, $Q$, or $V$?

$$\nabla U(\theta) = E_\tau \left[ \sum_{k=0}^{d} \nabla_\theta \log \pi_\theta(a_k \mid s_k) \gamma^k (r_k + \gamma V_\phi(s_{k+1}) - V_\phi(s_k)) \right]$$

temporal difference residual

$$l(\phi) = E\left[ \left( V_\phi(s) - V^{\pi_\theta}(s) \right)^2 \right]$$

↑ estimate with reward to go from sims

# Generalized Advantage Estimation

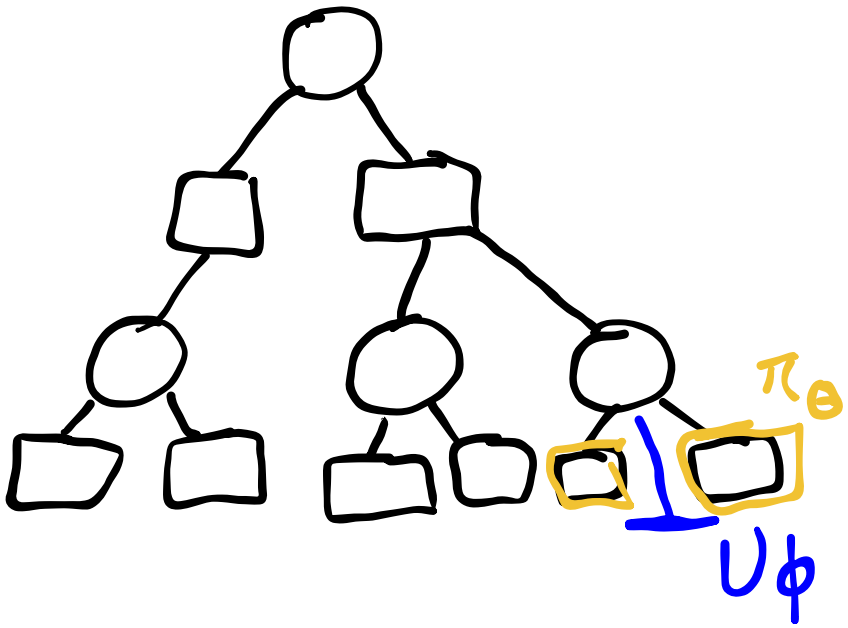
# Recap

# Alpha Zero: Actor Critic with MCTS

1. Use $\pi_\theta$ and $U_\phi$ in MCTS
2. Learn $\pi_\theta$ and $U_\phi$ from tree

$$\ell(\theta) = -\mathbb{E}_s\left[\sum_a \pi_{\text{MCTS}}(a \mid s) \log \pi_\theta(a \mid s)\right]$$

$$\pi_{\text{MCTS}}(a \mid s) \propto N(s,a)^\eta$$




$$\ell(\phi) = \frac{1}{2}\mathbb{E}_s\left[\left(U_\phi(s) - U_{\text{MCTS}}(s)\right)^2\right]$$

$$U_{\text{MCTS}}(s) = \max_a Q(s,a)$$

$$a = \arg\max_a Q(s,a) + c\pi_\theta(a \mid s)\frac{\sqrt{N(s)}}{1+N(s,a)}$$