Reinforcement Learning

Policy Gradient

Александр Костин
telegramm: @Ko3tin
LinkedIn: [kostinalexander]

Recap: Value-based Methods

Approximate action-value function with a neural network: $Q^*(s, a) \approx Q(s, a; \theta)$

Take action which maximises $Q(s, a; \theta)$

Recap: Value-based Methods

Approximate action-value function with a neural network: $Q^*(s, a) \approx Q(s, a; \theta)$

Take action which maximises $Q(s, a; \theta)$

+ Easy to generate policy
+ Close to true objective
+ Fairly well-understood, good algorithms exist
- Still not the true objective
- May focus capacity on irrelevant details
- Small value error can lead to larger policy error

Recap: Value-based Methods

Approximate action-value function with a neural network: $Q^*(s, a) \approx Q(s, a; \theta)$

Take action which maximises $Q(s, a; \theta)$

> "When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one."
>
> —Vladimir Vapnik

Recap: Objective

Suppose that since now we are living in the class of parametrised policies:

$\pi_\theta(a \mid s) = \mathbb{P}(A_t = a \mid S_t = s, \theta_t = \theta)$, where θ is some parameter.

$$\theta* = argmax_\theta \, J(\theta) = argmax_\theta \, \mathbb{E}_{p_\theta(\tau)}[\sum_{t=0}^{T} \gamma^t R_t] = argmax_\theta \, \mathbb{E}_{p_\theta(\tau)}[G(\tau)]$$

Objective

Suppose that since now we are living in the class of parametrised policies:

$\pi_\theta(a \mid s) = \mathbb{P}(A_t = a \mid S_t = s, \theta_t = \theta)$, where θ is some parameter.

$$\theta^* = argmax_\theta\, J(\theta) = argmax_\theta\, \mathbb{E}_{p_\theta(\tau)}[\sum_{t=0}^{T} \gamma^t R_t] = argmax_\theta\, \mathbb{E}_{p_\theta(\tau)}[G(\tau)]$$

$$J(\theta) = \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \int p_\theta(\tau) G(\tau) d\tau$$

Objective

Suppose that since now we are living in the class of parametrised policies:

$\pi_\theta(a \mid s) = \mathbb{P}(A_t = a \mid S_t = s, \theta_t = \theta)$, where θ is some parameter.

$$\theta^* = argmax_\theta\, J(\theta) = argmax_\theta\, \mathbb{E}_{p_\theta(\tau)}[\sum_{t=0}^{T} \gamma^t R_t] = argmax_\theta\, \mathbb{E}_{p_\theta(\tau)}[G(\tau)]$$

$$J(\theta) = \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \int p_\theta(\tau) G(\tau) d\tau$$

$$p_\theta(\tau) = p(s_0)\pi_\theta(a_0 \mid s_0) p(r_0, s_1 \mid s_0, a_0) \ldots$$

Objective's Gradient

$$p_\theta(\tau) = p(s_0)\pi_\theta(a_0 \,|\, s_0)p(r_0, s_1 \,|\, s_0, a_0)\ldots$$

$$\nabla J(\theta) = \nabla \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \int \nabla p_\theta(\tau) G(\tau) d\tau$$

Objective's Gradient

$$p_\theta(\tau) = p(s_0)\pi_\theta(a_0 \,|\, s_0)p(r_0, s_1 \,|\, s_0, a_0)\ldots$$

$$\nabla J(\theta) = \nabla \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \int \nabla p_\theta(\tau) G(\tau) d\tau$$

$$\nabla p_\theta(\tau) = p_\theta(\tau) \frac{\nabla p_\theta(\tau)}{p_\theta(\tau)} = p_\theta(\tau) \nabla \log p_\theta(\tau)$$

Objective's Gradient

$$p_\theta(\tau) = p(s_0)\pi_\theta(a_0 \mid s_0)p(r_0, s_1 \mid s_0, a_0)\ldots$$

$$\nabla J(\theta) = \nabla \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \int \nabla p_\theta(\tau) G(\tau) d\tau$$

$$\nabla p_\theta(\tau) = p_\theta(\tau) \frac{\nabla p_\theta(\tau)}{p_\theta(\tau)} = p_\theta(\tau) \nabla \log p_\theta(\tau) =$$

$$= p_\theta(\tau) \sum_{t=0}^{T} \nabla \log \pi_\theta(a_t \mid s_t)$$

Objective's Gradient

$$\nabla J(\theta) = \nabla \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \mathbb{E}_{p_\theta(\tau)}[\nabla \log p_\theta(\tau) G(\tau)]$$

REINFORCE (1992)

$$\nabla J(\theta) = \nabla \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \mathbb{E}_{p_\theta(\tau)}[\nabla \log p_\theta(\tau) G(\tau)]$$

Estimate gradient using Monte-Carlo estimator:

$$\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_{i,t} \mid s_{i,t}) G(\tau_i) \Big] = \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_{i,t} \mid s_{i,t}) \sum_{t=0}^{T} \gamma^t r_{i,t} \Big]$$

Make gradient ascent step:

$$\theta_{k+1} = \theta_k + \alpha \nabla J(\theta_k)$$

REINFORCE (1992)

$$\nabla J(\theta) = \nabla \mathbb{E}_{p_\theta(\tau)}[G(\tau)] = \mathbb{E}_{p_\theta(\tau)}[\nabla \log p_\theta(\tau) G(\tau)]$$

Estimate gradient using Monte-Carlo estimator:

$$\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_{i,t} | s_{i,t}) G(\tau_i)\Big] = \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_{i,t} | s_{i,t}) \sum_{t=0}^{T} \gamma^t r_{i,t}\Big]$$

Make gradient ascent step:

$$\theta_{k+1} = \theta_k + \alpha \nabla J(\theta_k)$$

Note that the algorithm is on-policy so old samples can not be used for gradient update $\longrightarrow$ No Replay Buffer

Connection with Behavioural Cloning

$$\nabla J_{BC}(\theta) = \mathbb{E}_{\tau \sim D}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t)\Big],$$

where D is a buffer contains samples collected by an expert

VS

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t) G(\tau)\Big]$$

Connection with Behavioural Cloning

$$\nabla J_{BC}(\theta) = \mathbb{E}_{\tau \sim D}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t)\Big],$$

Maximise log-likelihood (minimise cross-entropy loss) to take the similar actions as an expert.

where D is a buffer contains samples collected by an expert

VS

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t) G(\tau)\Big]$$

Learn actions which lead to higher returns

Entropy Regularisation

We would still like to sustain the exploration-exploitation trade-off.

Entropy Regularisation

We would still like to sustain the exploration-exploitation trade-off.

$$H(\pi_\theta(\,.\,|\,S_t)) = -\,\mathbb{E}_{\pi_\theta} \log \pi_\theta(\,.\,|\,S_t)$$ General case

$$H(\pi_\theta(\,.\,|\,S_t)) = -\sum_a \pi_\theta(a\,|\,S_t) \log \pi_\theta(a\,|\,S_t)$$ Discrete case

Entropy Regularisation

We would still like to sustain the exploration-exploitation trade-off.

$$H(\pi_\theta(\,.\,|\,S_t)) = -\,\mathbb{E}_{\pi_\theta} \log \pi_\theta(\,.\,|\,S_t) \quad \text{General case}$$

$$H(\pi_\theta(\,.\,|\,S_t)) = -\sum_a \pi_\theta(a\,|\,S_t) \log \pi_\theta(a\,|\,S_t) \quad \text{Discrete case}$$

Recall that uniform distribution has largest entropy while deterministic distribution has the lowest one.

We can add regularisation term $\rho H(\pi_\theta(\,.\,|\,S_t))$ to our objective:

- To encourage an agent to increase curiosity

Policy-based RL

+ Optimise true objective

+ Easy extended to high-dimensional or even continuous action spaced

+ Learn stochastic policies

+ No prior knowledge regarding the MDP dynamics

+ Sometimes it’s easy to learn policy directly instead of value function. Moreover, it seems more natural.

— Could get stuck in local optima

— Less sample efficient in comparison with value-based methods

— High variance

Policy-based RL

+ Optimise true objective

+ Easy extended to high-dimensional or even continuous action spaced

+ Learn stochastic policies

+ No prior knowledge regarding the MDP dynamics

+ Sometimes it's easy to learn policy directly instead of value function. Moreover, it seems more natural.

— Could get stuck in local optima

— Less sample efficient in comparison with value-based methods

— High variance ⟵ Let's decrease it

Variance Reduction

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) G(\tau)\Big] = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \boxed{\sum_{k=0}^{T} \gamma^k R_k}\Big]$$

Variance Reduction

Current action A_t influences only future rewards

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) G(\tau)\Big] = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \boxed{\sum_{k=0}^{T} \gamma^k R_k}\Big]$$

Variance Reduction

Current action $a_{i,t}$ influences only future rewards

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) G(\tau)\Big] = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \boxed{\sum_{k=0}^{T} \gamma^k R_k}\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \sum_{k=t}^{T} \gamma^k R_k\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \gamma^t \sum_{k=t}^{T} \gamma^{k-t} R_k\Big]$$

Variance Reduction

Current action $a_{i,t}$ influences only future rewards

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) G(\tau)\Big] = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \boxed{\sum_{k=0}^{T} \gamma^k R_k}\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \sum_{k=t}^{T} \gamma^k R_k\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \gamma^t \sum_{k=t}^{T} \gamma^{k-t} R_k\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \gamma^t Q_{\pi_\theta}(S_t, A_t)\Big]$$

Variance Reduction

Current action $a_{i,t}$ influences only future rewards

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) G(\tau)\Big] = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \boxed{\sum_{k=0}^{T} \gamma^k R_k}\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \sum_{k=t}^{T} \gamma^k R_k\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \gamma^t \sum_{k=t}^{T} \gamma^{k-t} R_k\Big]$$

$$= \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t) \gamma^t Q_{\pi_\theta}(S_t, A_t)\Big]$$

Let's ignore γ^t

Variance Reduction: Baseline

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t) Q_{\pi_\theta}(S_t, A_t)\Big]$$

Consider some baseline $b(S_t)$ and compute $\mathbb{E}_{p_\theta(\tau)}\big[b(S_t)\, \nabla \log \pi(A_t \,|\, S_t)\big]$:

Variance Reduction: Baseline

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t) Q_{\pi_\theta}(S_t, A_t)\Big]$$

Consider some baseline $b(S_t)$ and compute $\mathbb{E}_{p_\theta(\tau)}\left[b(S_t)\,\nabla \log \pi(A_t \,|\, S_t)\right]$:

$$\mathbb{E}_{p_\theta(\tau)}\left[b(S_t)\,\nabla \log \pi(A_t \,|\, S_t)\right] = \int p_\theta(\tau) b(S_t)\,\nabla \log p_\theta(\tau) d\tau =$$

$$= \int b(S_t)\,\nabla p_\theta(\tau) d\tau = b(S_t)\,\nabla\,\mathbb{E}_{p_\theta(\tau)}[1] = b(S_t)\,\nabla\,[1] = 0$$

$$Var(Q(s,a) - b(s)) = Var(Q(s,a)) + Var(b(s)) - 2cov(Q(s,a), b(s))$$

Variance Reduction: Baseline

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t)[Q_{\pi_\theta}(S_t, A_t) - b(S_t)]\Big]$$

Variance Reduction: Baseline

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t)[Q_{\pi_\theta}(S_t, A_t) - b(S_t)]\Big]$$

Typically we take $V_{\pi_\theta}(S_t)$ as a baseline so $Q_{\pi_\theta}(S_t, A_t) - V_{\pi_\theta}(S_t) = A_{\pi_\theta}(S_t, A_t)$ is an advantage function considered in the previous lecture among DQN modification.

Actor-Critic

$$\nabla J_{AC}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \mid S_t)[A_{\pi_\theta}(S_t, A_t)]\Big]$$

We can approximate $A_{\pi_\theta}(S_t, A_t)$ with a neural network $A(S_t, A_t; \phi)$

… but we can make slightly better

Actor-Critic

$$\nabla J_{AC}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t)[A_{\pi_\theta}(S_t, A_t)]\Big]$$

We can approximate $A_{\pi_\theta}(S_t, A_t)$ with a neural network $A(S_t, A_t; \phi)$

… but we can make slightly better

$$A_{\pi_\theta}(s, a) = Q_{\pi_\theta}(s, a) - V_{\pi_\theta}(s) = \mathbb{E}_{r,s' \sim p(.|s,a)}[r + \gamma V(s')] \approx r + \gamma V_{\pi_\theta}(s') - V_{\pi_\theta}(s)$$

for the transition (s, a, r, s')

Advantage Actor-Critic

- Generate trajectories $\{\tau_i\}$ following $\pi_\theta(a \mid s)$
- Policy improvement:

  Estimate gradient and make gradient ascent step:

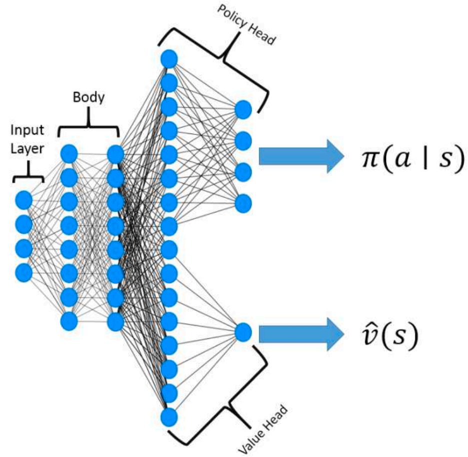
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_{i,t} \mid s_{i,t}) A_{\pi_\theta}(s_{i,t}, a_{i,t}) \right]$$

- Policy evaluation:

  Estimate gradient and make gradient descent step:

$$\nabla_\phi L(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=0}^{T} \nabla_\phi (r_{i,t} + \gamma \boxed{V_{\phi^-}}(s_{i,t+1}) - V_\phi(s_{i,t}))^2 \right]$$

Not target network, just frozen parameters



Source

Asynchronous Advantage Actor-Critic (A3C)

Asynchronous Methods for Deep Reinforcement Learning

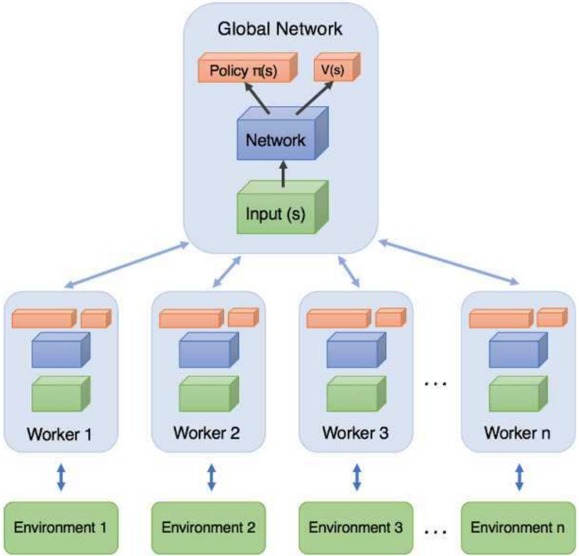
Volodymyr Mnih[1] VMNIH@GOOGLE.COM
Adrià Puigdomènech Badia[1] ADRIAP@GOOGLE.COM
Mehdi Mirza[1,2] MIRZAMOM@IRO.UMONTREAL.CA
Alex Graves[1] GRAVESA@GOOGLE.COM
Tim Harley[1] THARLEY@GOOGLE.COM
Timothy P. Lillicrap[1] COUNTZERO@GOOGLE.COM
David Silver[1] DAVIDSILVER@GOOGLE.COM
Koray Kavukcuoglu[1] KORAYK@GOOGLE.COM

[1] Google DeepMind
[2] Montreal Institute for Learning Algorithms (MILA), University of Montreal

Original paper

A3C

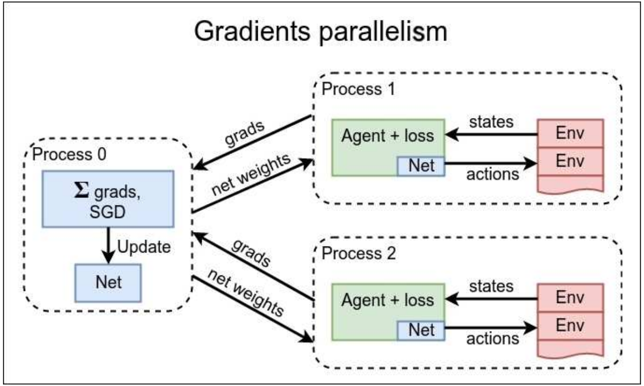
- N-step advantage estimation
- LSTM network
- No experience replay
- Entropy regularisation



Source

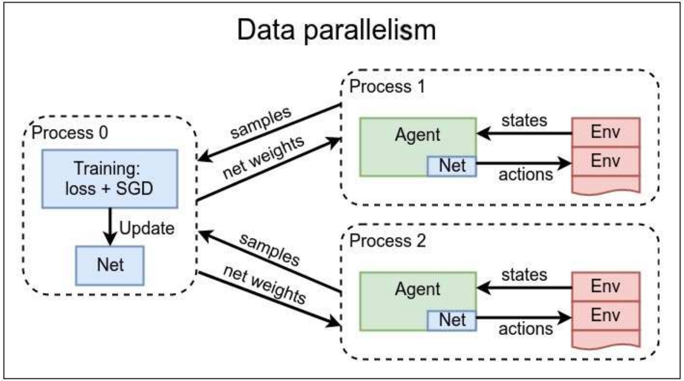
Asynchronous vs Parallel

A3C



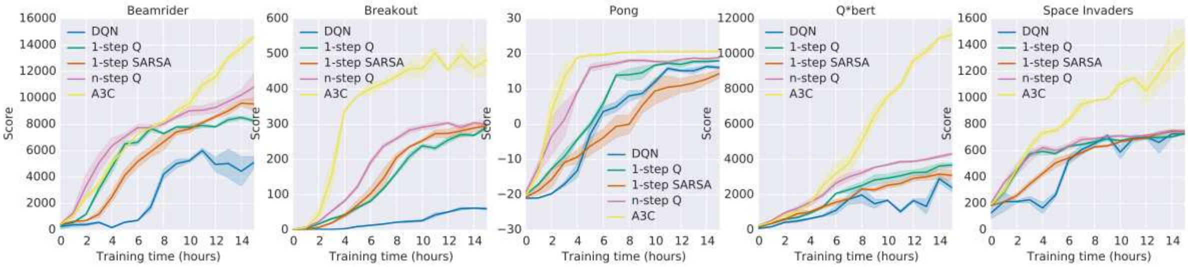
Source

A2C



Source

Comparison



Method	Training Time	Mean	Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

Table 1. Mean and median human-normalized scores on 57 Atari games using the human starts evaluation metric. Supplementary Table SS3 shows the raw scores for all games.

Source