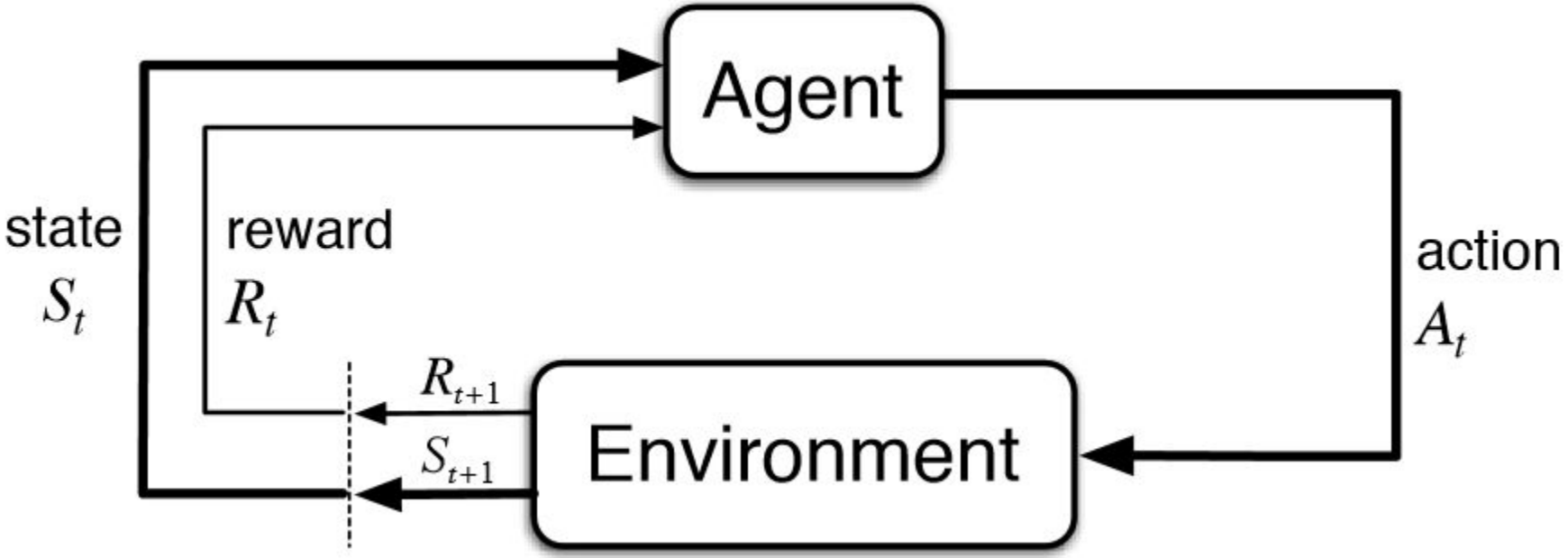
StarAi: Deep Reinforcement Learning

Overview

- Recap
- Rainbow Overview
- Double DQN
- Dueling DQN
- Multistep DQN
- Prioritised Experience Replay
- Noisy + Distribution (overview)
- PYSC2

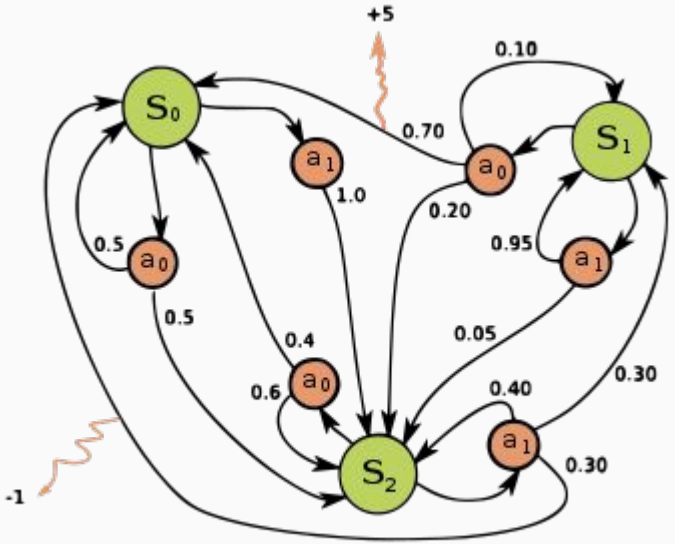
RL environment



* Image from taken from somewhere on the web :)

Recap

- Epsilon → Greedy
- Markov Decision Processes



* Image from taken from wikipedia

Recap

- Q Learning (simplified)

$$Q(S_t, A_t) \leftarrow R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$$

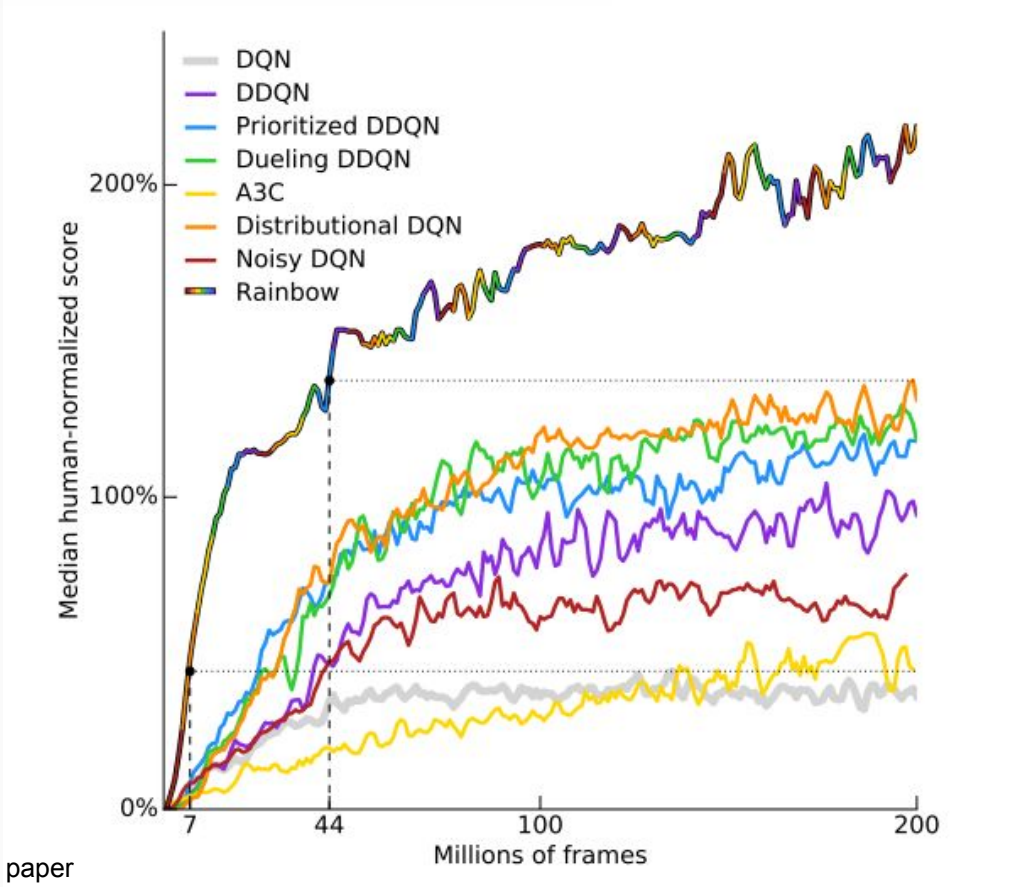
- DQN

$$Q(S_t, A_t, \theta) \leftarrow R_{t+1} + \gamma \max_a Q(S_{t+1}, a, \theta_{target})$$

$$L(\theta) = Q(S_t, A_t, \theta) - [R_{t+1} + \gamma \max_a Q(S_{t+1}, a, \theta_{target})]$$

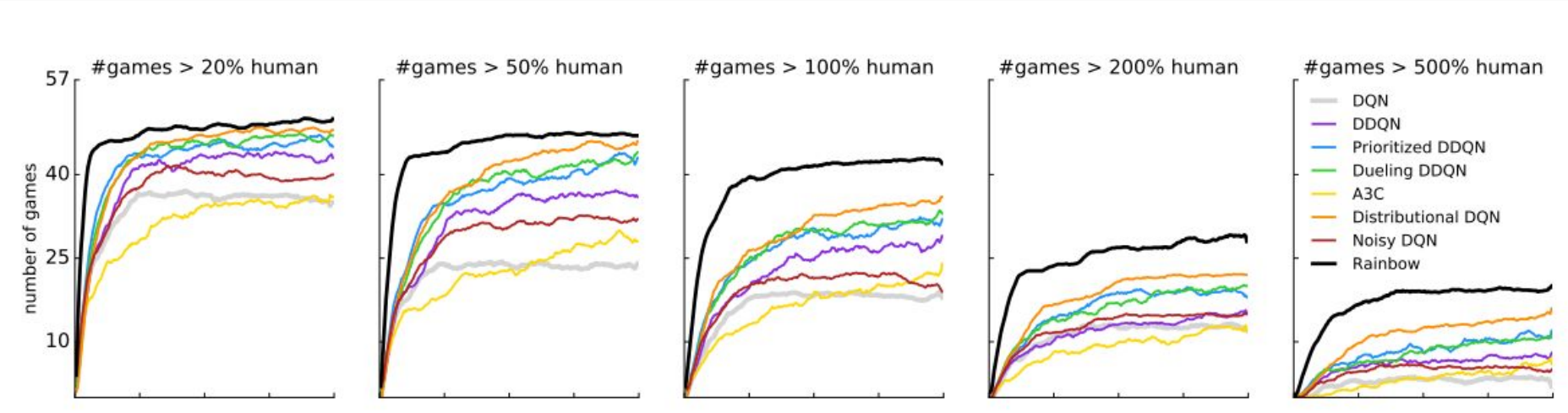
Rainbow

Rainbow on Atari



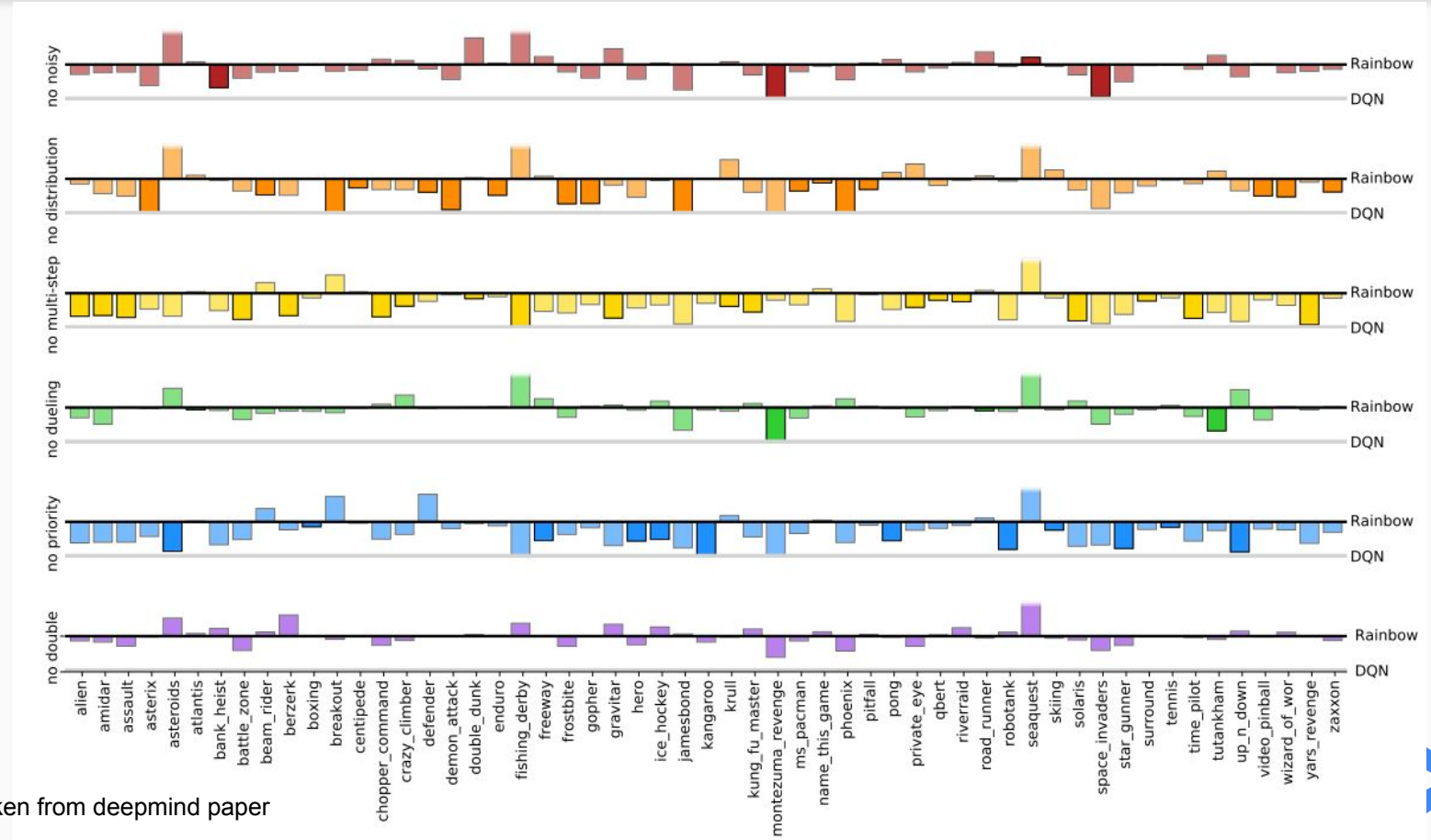
* Image from taken from deepmind paper

Rainbow on Atari



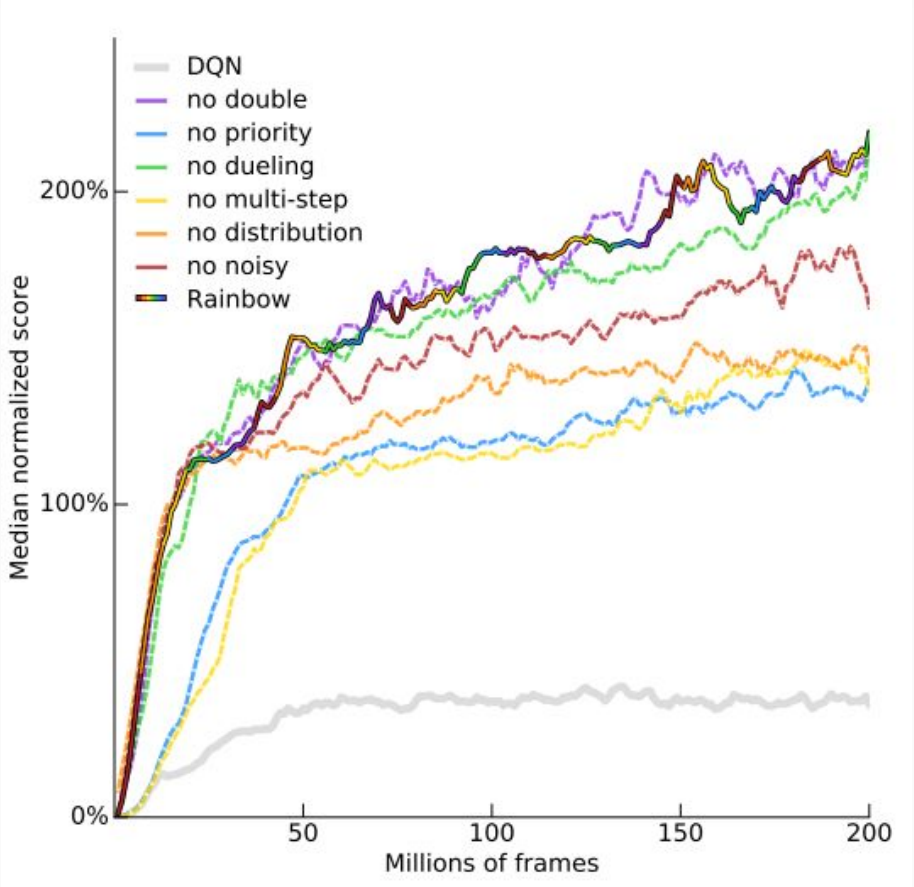
* Image from taken from deepmind paper

Rainbow Ablations



* Image from taken from deepmind paper

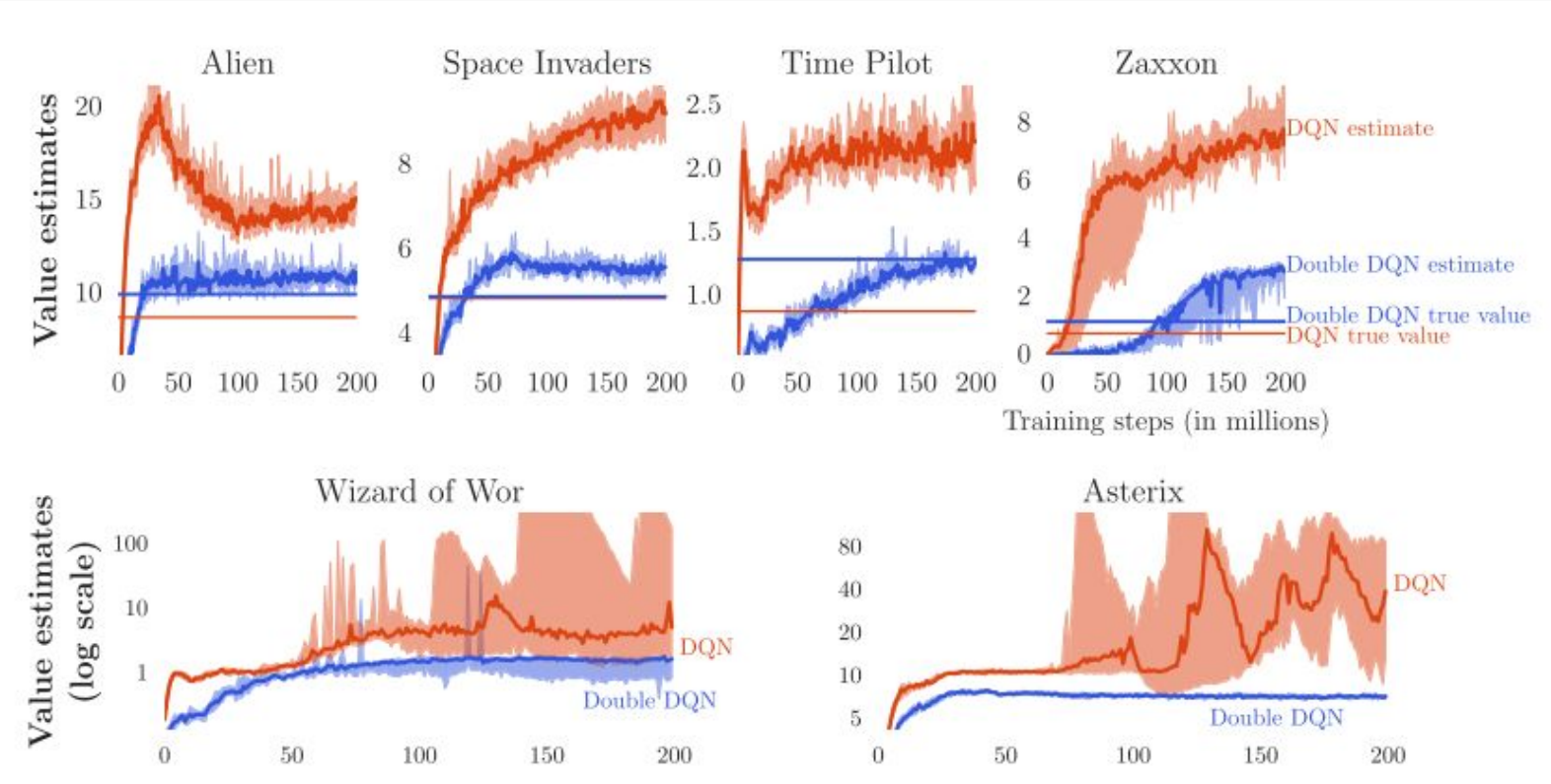
Rainbow Ablations



* Image from taken from deepmind paper

Double DQN

Double DQN



* Image from taken from paper

Double DQN

DQN

$$Q(S_t, A_t, \theta) \leftarrow R_{t+1} + \gamma \max_a Q(S_{t+1}, a, \theta_{target})$$

$$L(\theta) = Q(S_t, A_t, \theta) - y \text{ where } y = [R_{t+1} + \gamma \max_a Q(S_{t+1}, a, \theta_{target})]$$

Double Q

everything is the same except for

$$y = [R_{t+1} + \gamma\, Q(S_{t+1}, \max_a Q(S_{t+1}, a, \theta), \theta_{target})]$$

Link to notebooks

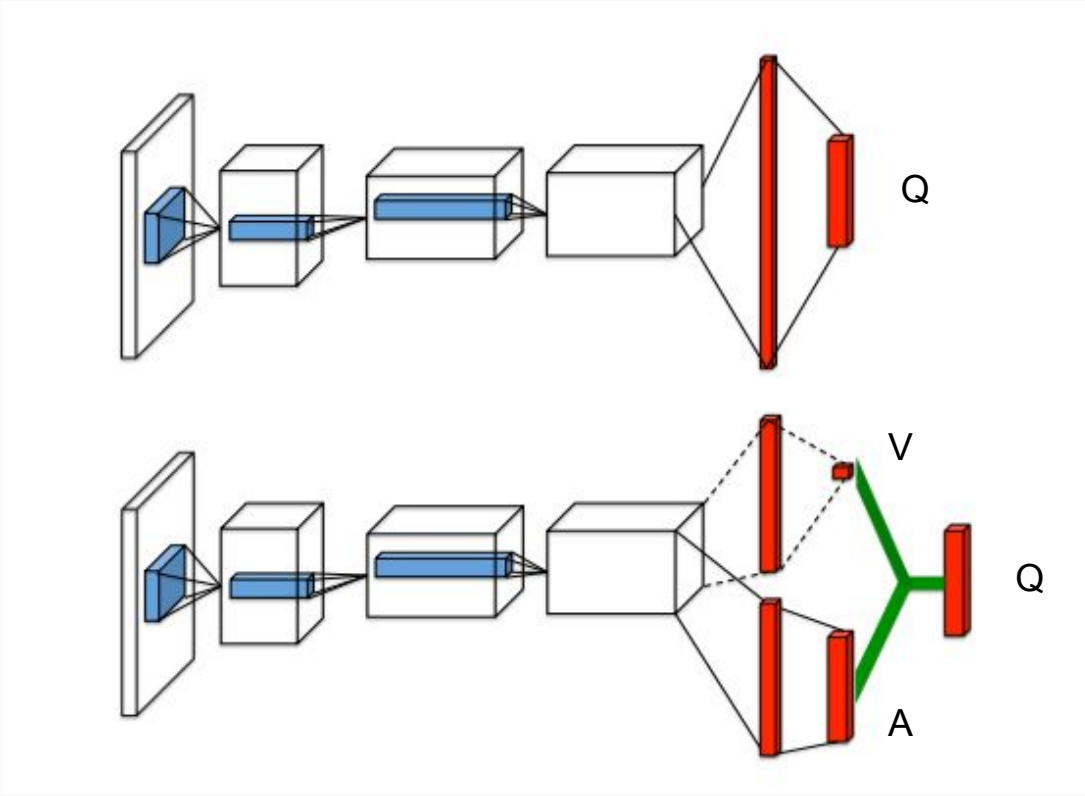
https://bit.ly/2srCfoH

Dueling DQN

Dueling DQN

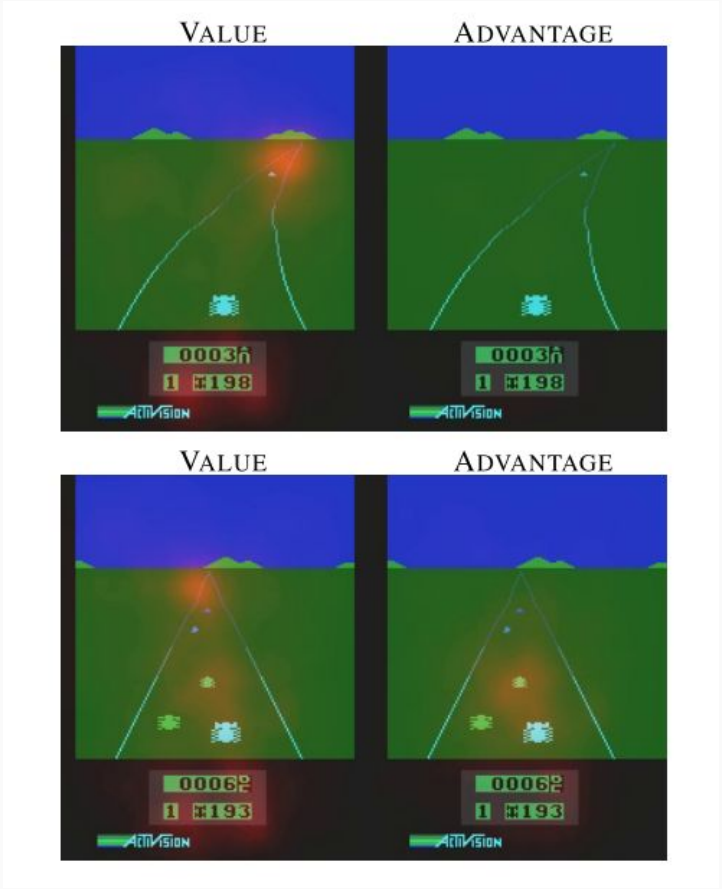
DQN

Dueling



* Image from taken from paper

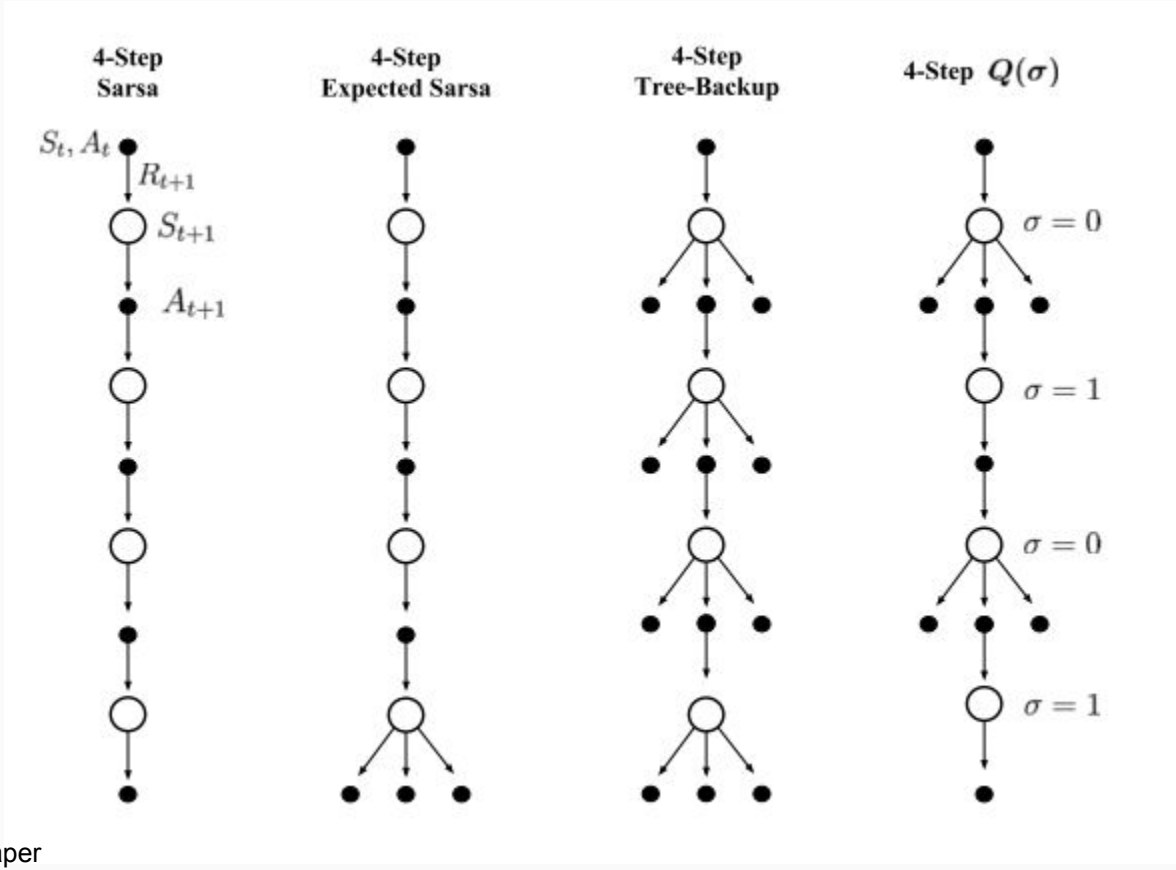
Dueling DQN



* Image from taken from paper

Multistep

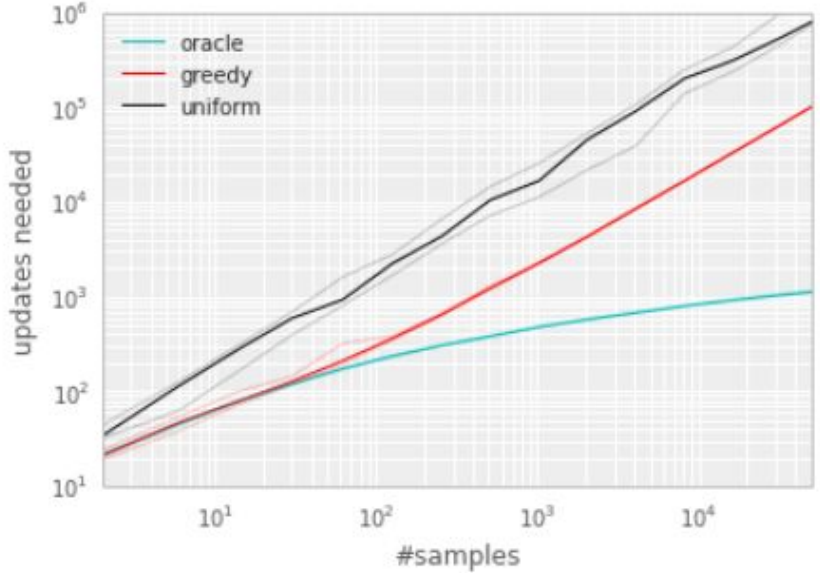
Multistep DQN

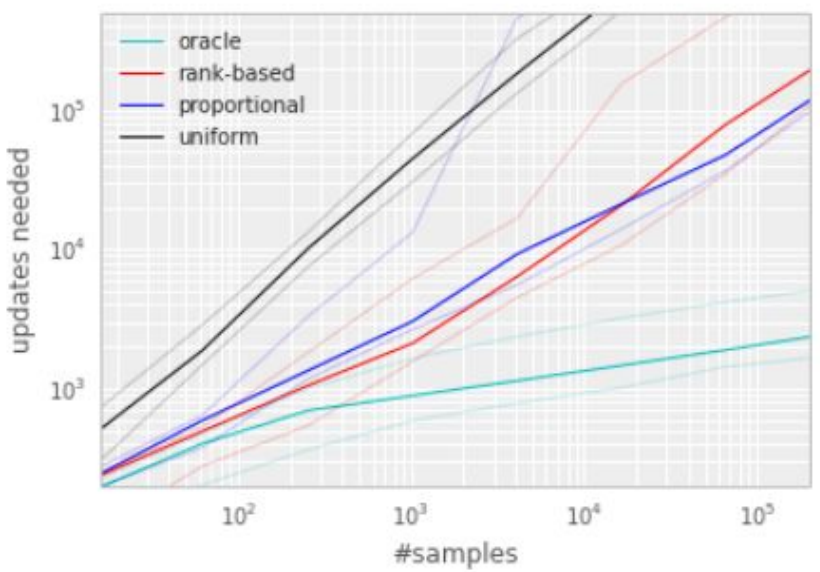


* Image from taken from paper

Prioritised Experience Replay

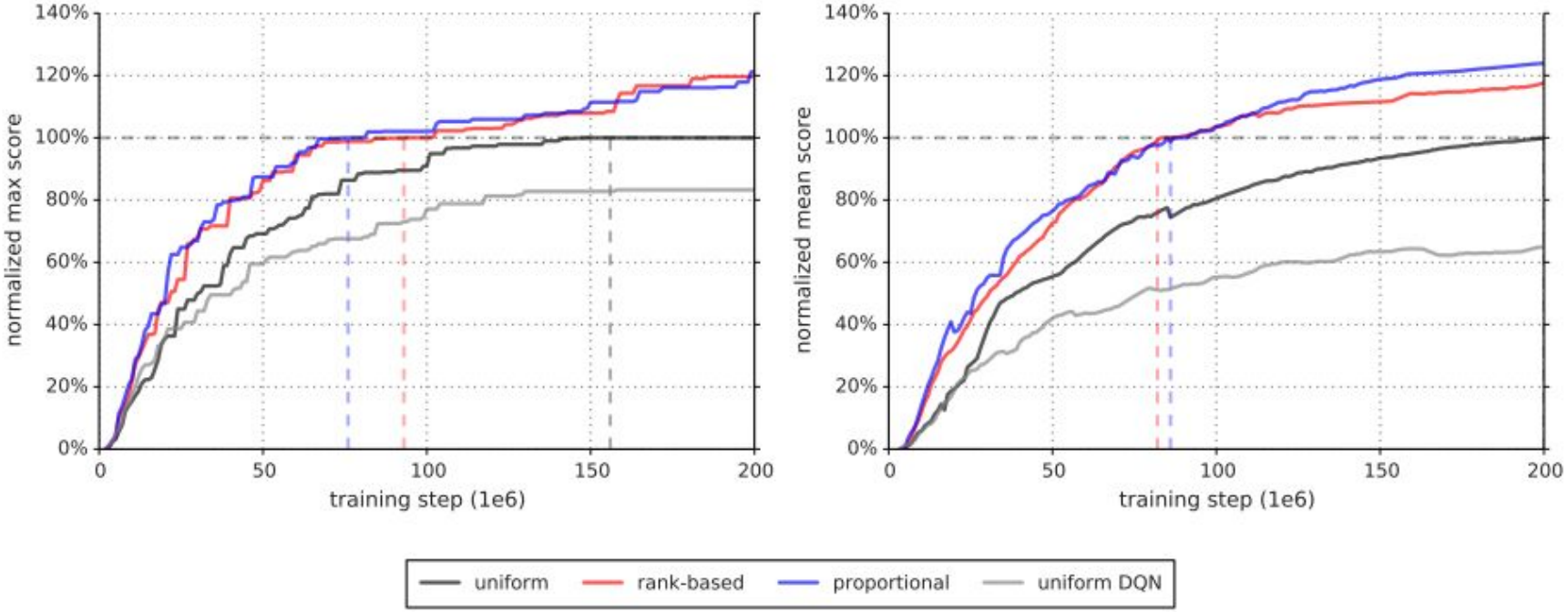
Prioritised Experience Replay - Blind Cliffwalk





* Image from taken from paper

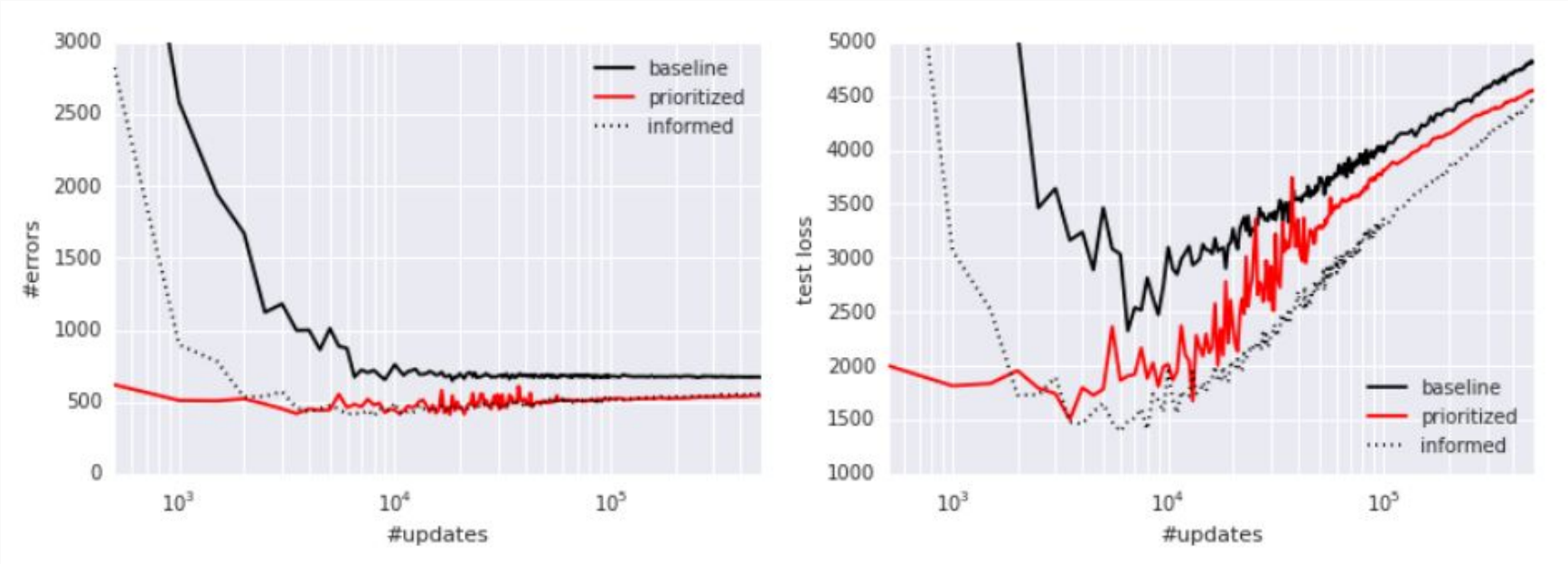
Prioritised Experience Replay - Atari Games



* Image from taken from paper

Prioritised Experience Replay - Supervised Learning

- MNIST
- Removed 99% of samples of 0,1,2,3,4



* Image from taken from paper

Noisy + Distribution

PYSC2

Starcraft 2

- Real time strategy game released in July 2010 by Blizzard and continuously updated
- 3 unique races requiring different tactics to win
- Economy, base management, army management
- Fog of war, exploration, multi tasking and micro management

Watch a match segment

Mini Games

- Move to beacon
- Collect Mineral Shards
- Find and defeat zergling
- Defat roaches
- Defeat zerglings and banelings
- Collect minerals and gas
- Build marines

Observation Space

Minimap feature layers:

- **height_map**: Shows the terrain levels.
- **visibility**: Which part of the map are hidden, have been seen or are currently visible.
- **creep**: Which parts have zerg creep.
- **camera**: Which part of the map are visible in the screen layers.
- **player_id**: Who owns the units, with absolute ids.
- **player_relative**: Which units are friendly vs hostile. Takes values in [0, 4], denoting [background, self, ally, neutral, enemy] units respectively.
- **selected**: Which units are selected.

Observation Space

Screen feature layers:

- **height_map**: Shows the terrain levels.
- **visibility**: Which part of the map are hidden, have been seen or are currently visible.
- **creep**: Which parts have zerg creep.
- **power**: Which parts have protoss power, only shows your power.
- **player_id**: Who owns the units, with absolute ids.
- **player_relative**: Which units are friendly vs hostile. Takes values in [0, 4], denoting [background, self, ally, neutral, enemy] units respectively.

Observation Space

Screen feature layers:

- **unit_type**: A unit type id, which can be looked up in pysc2/lib/units.py.
- **selected**: Which units are selected.
- **hit_points**: How many hit points the unit has.
- **energy**: How much energy the unit has.
- **shields**: How much shields the unit has. Only for protoss units.
- **unit_density**: How many units are in this pixel.
- **Unit_density_aa**: Anti-aliased

Observation Space

Structured layers:

- **General Player Information**: player_id, food, minerals etc
- **Control Groups**: saved unit group info
- **Multi Select**: info about selected unit
- **Cargo**: Units in a transport
- **Build Queue**: Units in production
- **Available actions**: all actions made successfully since last observation
- **Action Result**: result of action
- **Alerts**: whether you are being attacked

Action Space

See it in pycharm