MLDS 2019 Spring
HW4-1 - Policy Gradient

2018/05/25

adlxmlds@gmail.com

Time Schedule

- June 4th 19:30 Deadline
- June 8th 10:30 Deadline

Outline

	HW4-1	HW4-2	HW4-3
Pong	PG		AC
Breakout		DQN	AC
Improved Version			

Outline

- **Introduction**
    - Game Playing : Pong
- **Deep Reinforcement Learning**
    - Policy Gradient
    - Improvements to Policy Gradient
- **Training Hints**
- **Grading & Format**
    - Submission

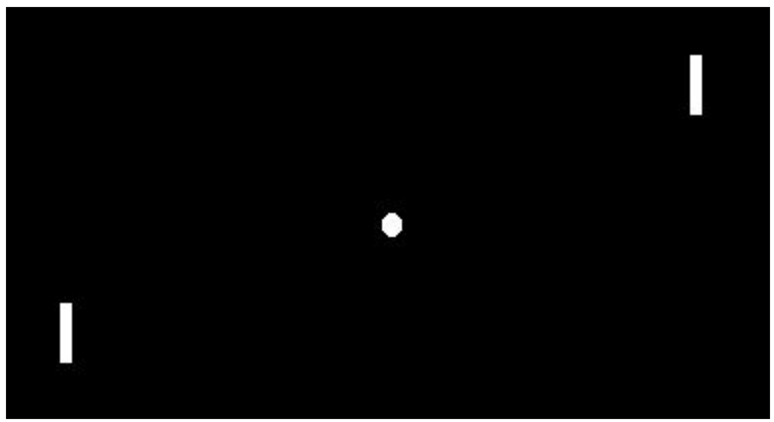
Introduction

Game Playing

- Implement an agent to play Atari games using Deep Reinforcement Learning
- In this homework, you are required to implement **Policy Gradient**
- The Pong environment is used in this homework

Environment

Pong



https://gym.openai.com/envs/

Deep Reinforcement Learning

Policy Gradient

function REINFORCE
  Initialise θ arbitrarily
  for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
    end for
  end for
  return θ
end function

s_i : state at time i
a_i : action at time i
r_i : reward by a_i
$\pi_\theta(s, a) = P[a|s, \theta]$: θ is your model parameter
v_t : long-term value at time t
$v(s) = E[\ G_t|s_t = s]$
$G_t = \sum_{k=0}^{\text{inf}} \gamma^k r_{t+k+1}$

- Update per step → SGD → High Variance
- Update per episode or by mini batch
  - episode : A player win the game (21)
  - mini batch : someone get some points

REINFORCE Baseline on Pong

Training loop(simplest version):

a. Play until a game is over(one player gets 21 points) with policy network π_θ and store (s,a,r) tuples into memory m.
b. Discount and normalize rewards in memory into r' to reduce variance
c. Approximate gradient $\nabla_\theta J(\theta) \approx \sum_{(s_t, a_t, r'_t) \in m} \nabla_\theta \log \pi_\theta(a_t|s_t) r'_t$

   $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
d. Clear the memory m

Improvements to Policy Gradient

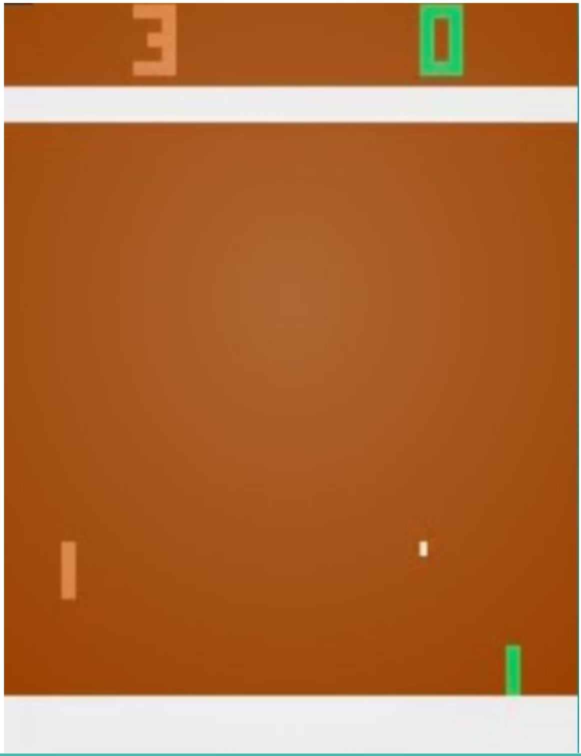
- Variance Reduction
- Natural Policy Gradient
- Trust Region Policy Optimization
- Proximal Policy Optimization

http://rll.berkeley.edu/deeprlcourse/f17docs/lecture_4_policy_gradient.pdf
http://rll.berkeley.edu/deeprlcourse/f17docs/lecture_13_advanced_pg.pdf

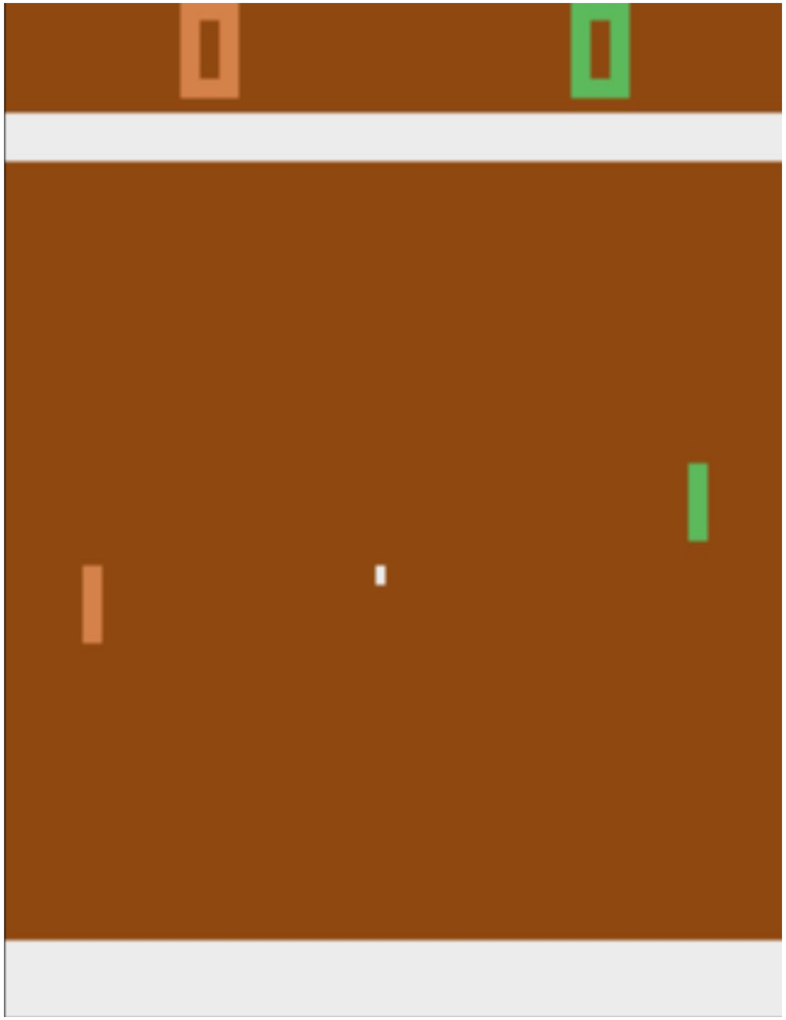
Training Hint

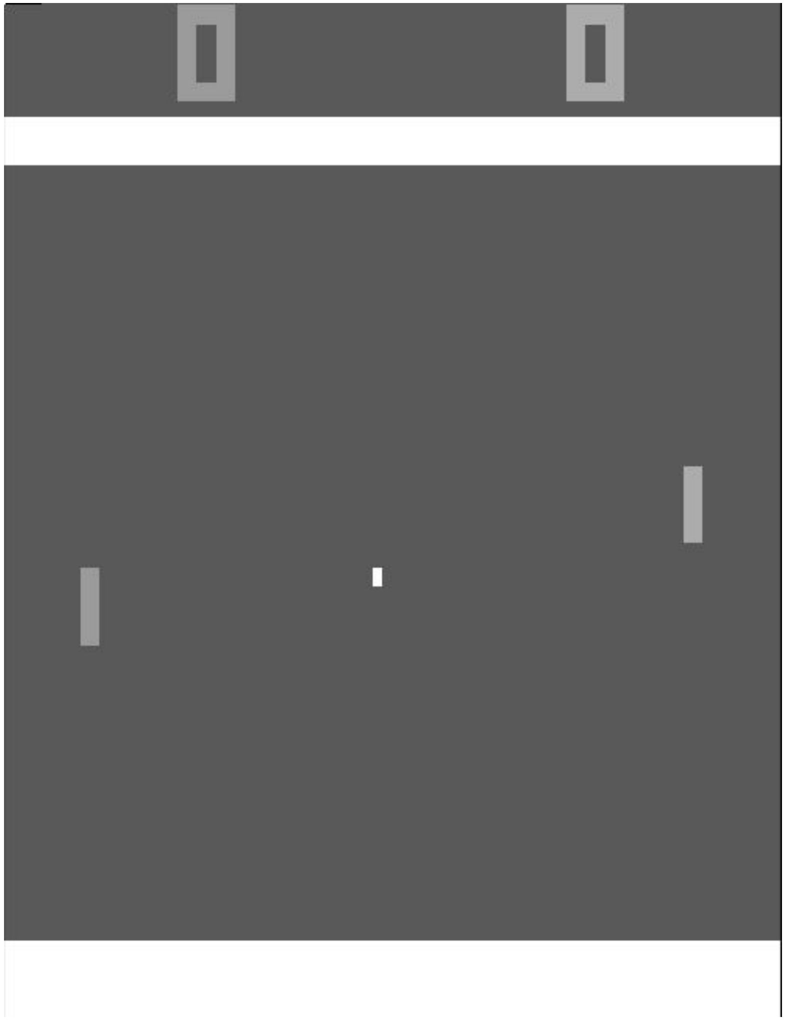
Preprocessing for States

- Which is better ?
  - rgb channel or gray scale
    - 0.2126 * Red + 0.7152 * Green + 0.0722 * Blue
  - single or residual
    - s'(t) = s(t+1) - s(t)
    - represent change of pixel
  - scoreboard yes or no?

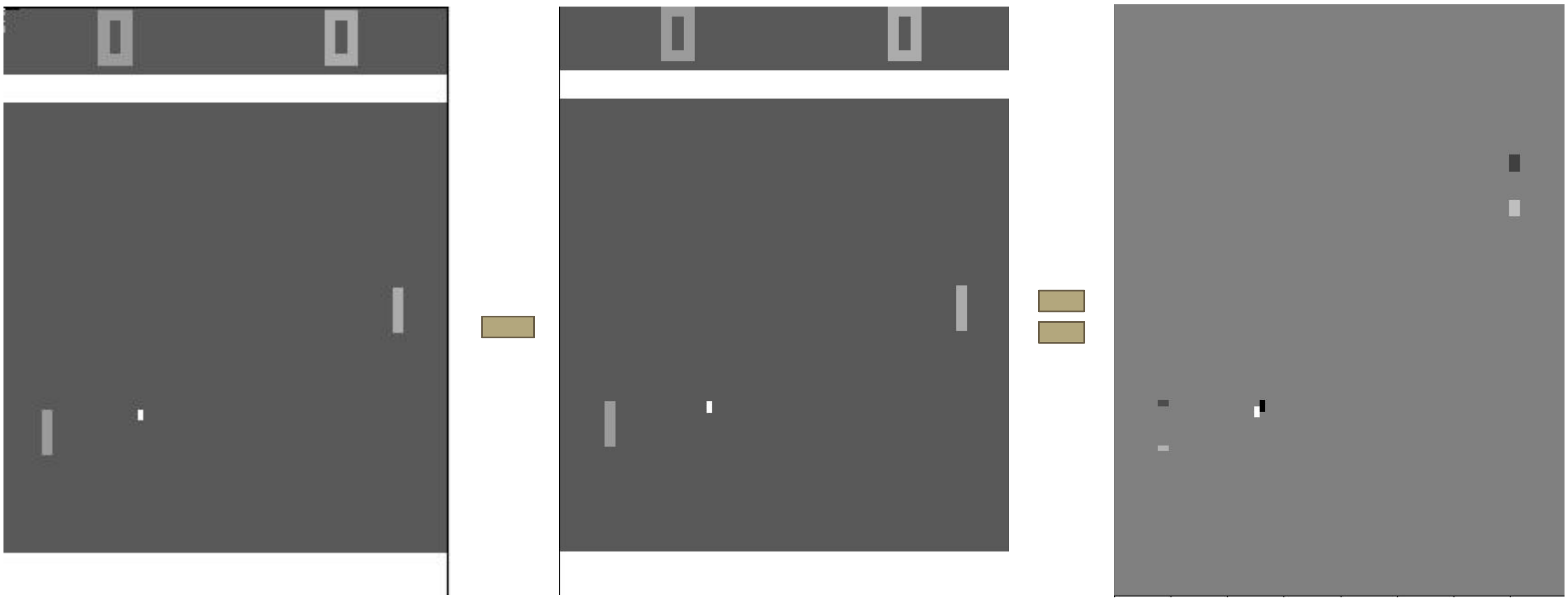


RGB vs Gray scale





Residual State



S2 S1 S1'

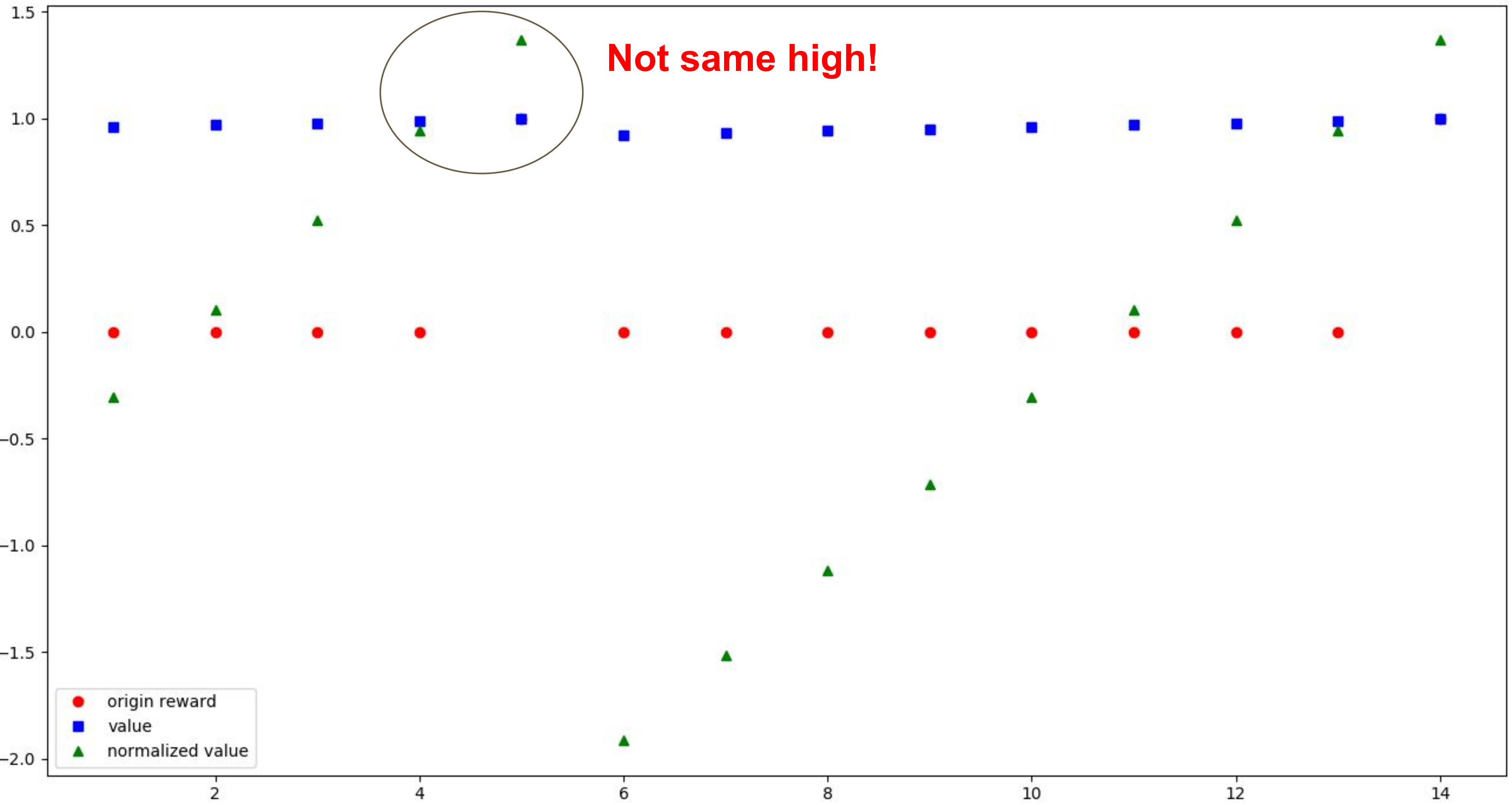
Reward and Action

- Reward normalization
  - More stable
  - http://karpathy.github.io/2016/05/31/rl
  - https://arxiv.org/pdf/1506.02438.pdf
- Action space reduction
- Reset the running add of discounted reward to zero if a player scores (Pong specific)

Action space reduction

```
ACTION_MEANING = {
    0: "NOOP",
    1: "FIRE",
    2: "UP",
    3: "RIGHT",
    4: "LEFT",
    5: "DOWN",
    6: "UPRIGHT",
    7: "UPLEFT",
    8: "DOWNRIGHT",
    9: "DOWNLEFT",
    10: "UPFIRE",
    11: "RIGHTFIRE",
    12: "LEFTFIRE",
    13: "DOWNFIRE",
    14: "UPRIGHTFIRE",
    15: "UPLEFTFIRE",
    16: "DOWNRIGHTFIRE",
    17: "DOWNLEFTFIRE",
}
```

Reward normalization

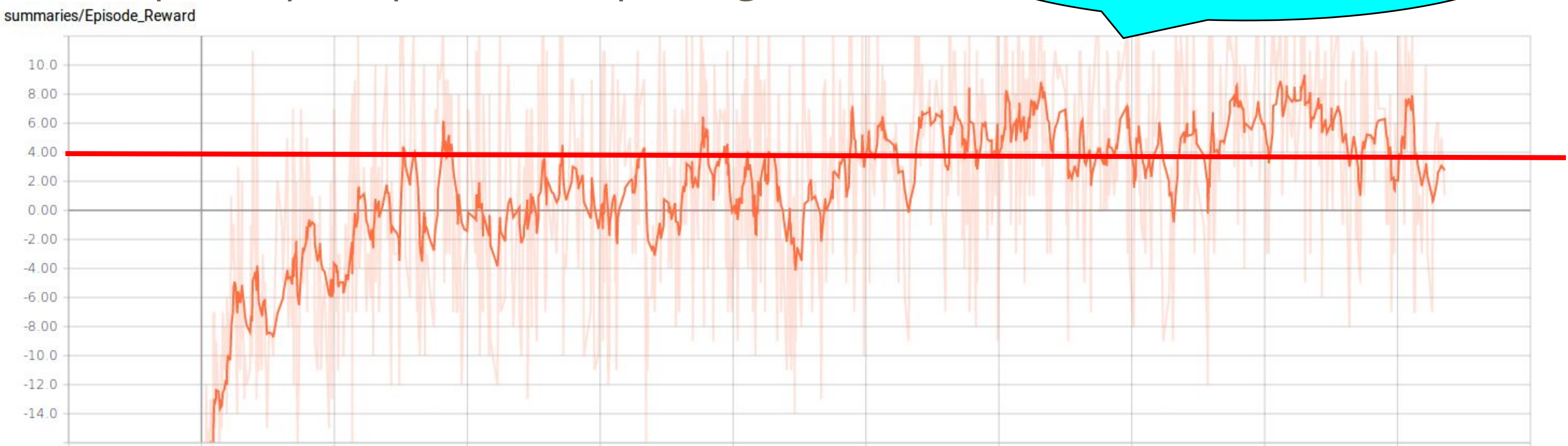


Training Hint

Training Plot

- The unit of x-axis is 1000 episode
- Around 6000 episode to reach baseline in “average”
- Mind your preprocessing if your curve differs from this too much
- Baseline Network Structure : Flatten + Two-layer FNN
    - 256 dimension hidden layer
    - output layer (action space size)
- Update per episode (21 point game)

Freeze random seed!



Grading & Format

Baseline

- Policy Gradient
  - Getting averaging reward in 30 episodes over **0** in **Pong**
  - Without OpenAI’s Atari wrapper & reward clipping
  - Improvements to Policy Gradient are allowed

Grading & Format

Code Format

- Please download the sample files from github
- Follow the instructions in README to install required packages
- **Four** functions you should implement in agent_pg.py
    1. __init__(self, env, args)
    2. init_game_setting(self)
    3. train(self)
    4. make_action(self, state, test)
- **DO NOT** add any parameter in __init__(), init_game_setting() and make_action()
- You can add new methods in the agent_pg.py
- You can add your arguments in argument.py

Grading & Format

Submission

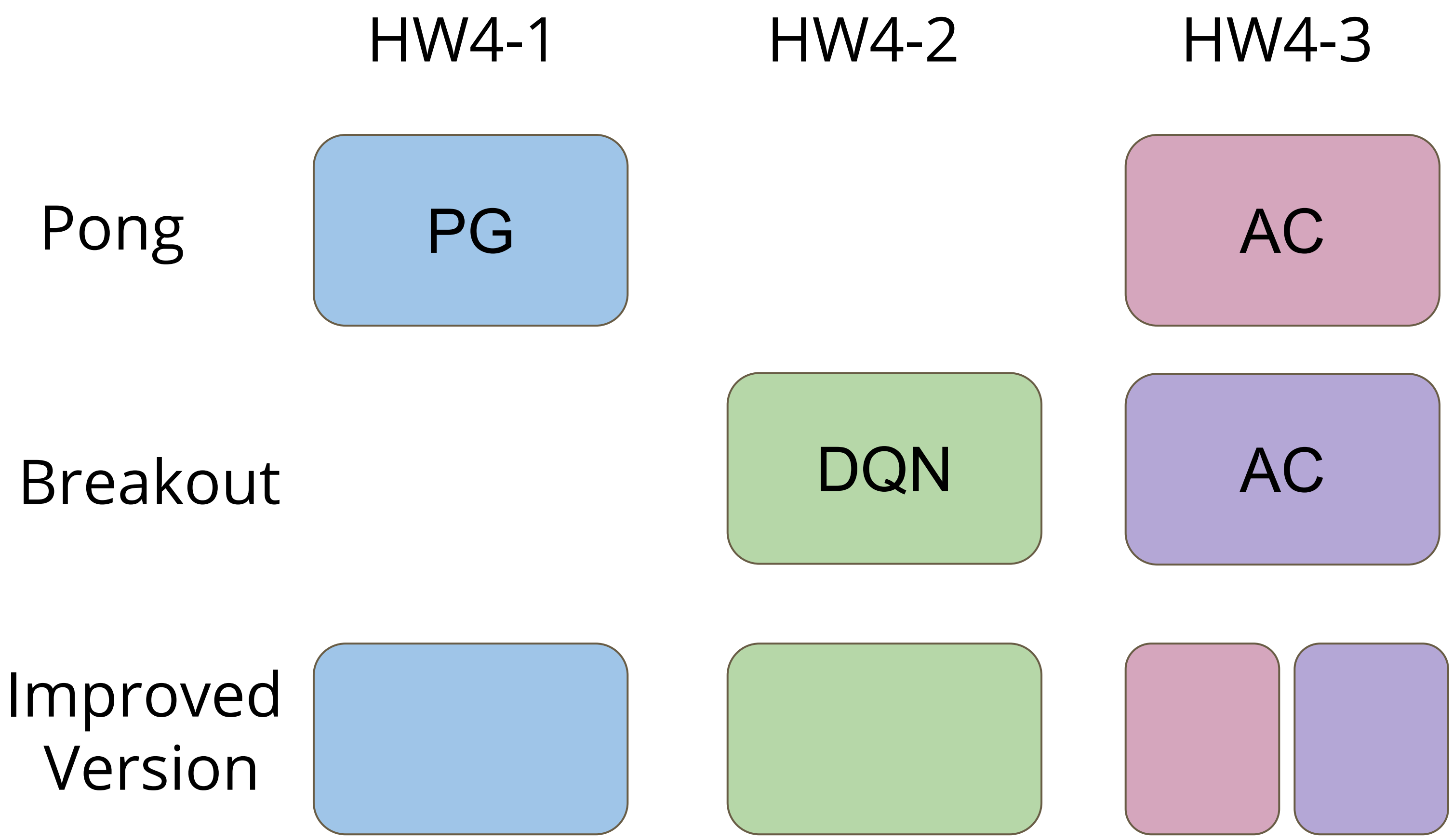
- Submit your presentation files to: [google drive]

Grading & Format

Presentation Files

- Describe your Policy Gradient model
- Plot the learning curve to show the performance of your Policy Gradient on Pong
    - X-axis: number of time steps
    - Y-axis: average reward in last 30 episodes
- Implement 1 improvement method on page 8
    - Describe your tips for improvement
    - Learning curve
    - Compare to the vanilla policy gradient
- Github link

Outline



Related Materials

- Course & Tutorial:
  - Berkeley Deep Reinforcement Learning, Fall 2017
  - David Silver RL course
  - Nips 2016 RL tutorial
- Blog:
  - Andrej Karpathy's blog
  - Arthur Juliani's Blog
  - Openai
- Text Book:
  - Reinforcement Learning: An Introduction
- Repo:
  - https://github.com/williamFalcon/DeepRLHacks