Paradigmns of Reinforcement Learning

Final Project

Maria Pilligua and Nil Biescas

Overview

01 ‘’Simple’ environment: Freeways

02 Complex environment: Tennis

03 Pong World Tournament

Simple Environment: Freeways

Description of the env

Observation Space

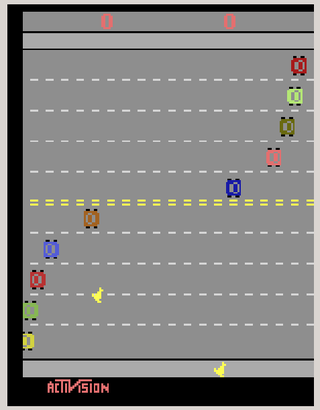
Action Space	Discrete(3)
Observation Space	Box(0, 255, (210, 160, 3), uint8)

Available Actions

Value	Meaning	Value	Meaning	Value	Meaning
0	NOOP	1	UP	2	DOWN

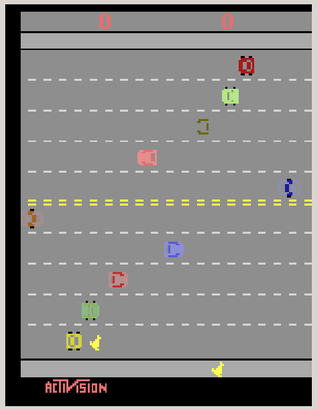
Pre-Processing

Original



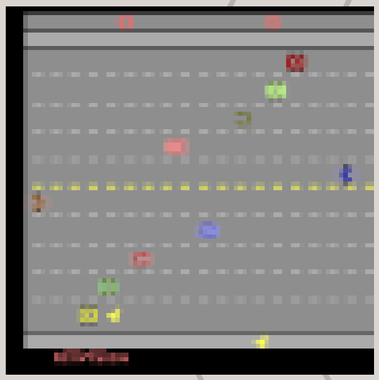
(210, 160, 3)

MaxAndSkip



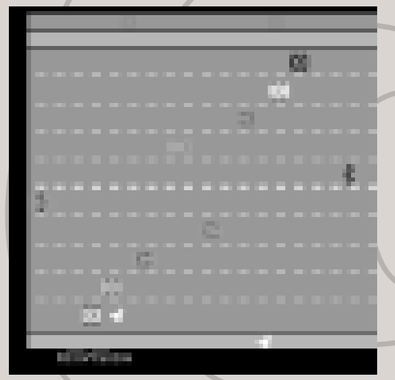
(210, 160, 3)

Resize



(84, 84, 3)

Grayscale



(84, 84, 1)

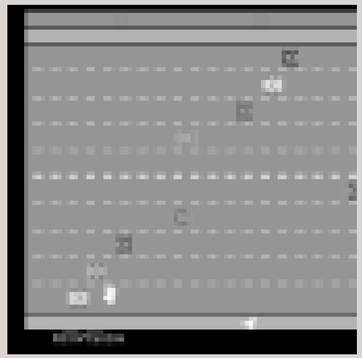
Reshape

(84, 84)

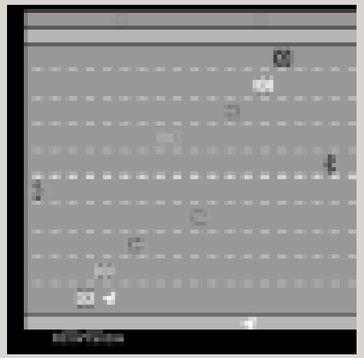
ImageToPyTorch

(1, 84, 84)

FrameStack

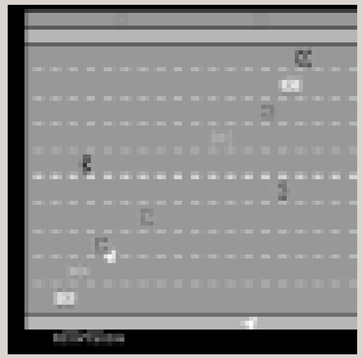


Frame 1

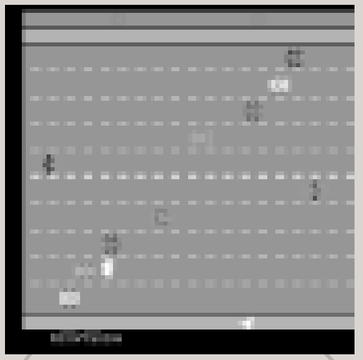


Frame 2

(4, 84, 84)



Frame 3



Frame 4

ScaledFloatFrame

(4, 84, 84)

Models trained

DQN (+ extensions): value-based.

Reinforce: policy-based

Models trained

DQN (+ extensions): value-based

- **Prioritized Replay Buffer**
- **Dueling DQN**
- **Double DQN**
- **Noisy DQN**
- **Two-step DQN**

Reinforce: policy-based

Models trained

DQN (+ extensions): value-based

- **Prioritized Replay Buffer:** using TD error, give **priority** to “important” experiences.
- **Dueling DQN**
- **Double DQN**
- **Noisy DQN**
- **Two-step DQN**

Reinforce: policy-based

Models trained

DQN (+ extensions): value-based

- **Prioritized Replay Buffer:** using TD error, give **priority** to “important” experiences.
- **Dueling DQN: Separate** Q-value estimation into state-value and advantage.
- **Double DQN**
- **Noisy DQN**
- **Two-step DQN**

Reinforce: policy-based

Models trained

DQN (+ extensions): value-based

- **Prioritized Replay Buffer:** using TD error, give **priority** to “important” experiences.
- **Dueling DQN:** **Separate** Q-value estimation into state-value and advantage.
- **Double DQN:** Reduce overestimation using **a net for action selection** and another for Q-value estimation.
- **Noisy DQN**
- **Two-step DQN**

Reinforce: policy-based

Models trained

DQN (+ extensions): value-based

- **Prioritized Replay Buffer:** using TD error, give **priority** to “important” experiences.
- **Dueling DQN:** **Separate** Q-value estimation into state-value and advantage.
- **Double DQN:** Reduce overestimation using **a net for action selection** and another for Q-value estimation.
- **Noisy DQN:** Introduce **noise** into the network **weights** to encourage exploration.
- **Two-step DQN**

Reinforce: policy-based

Models trained

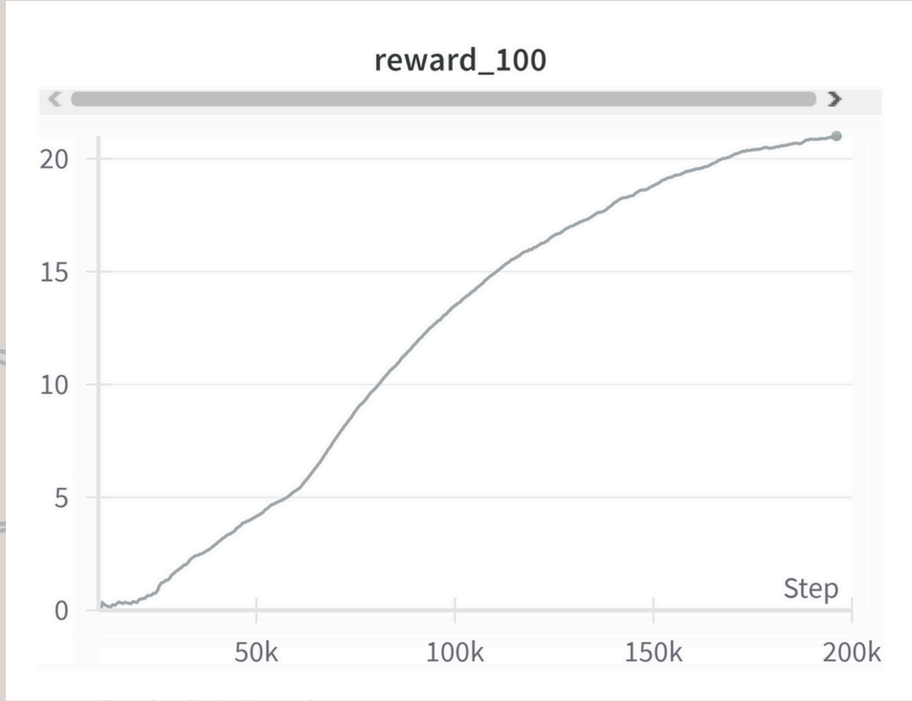
DQN (+ extensions): value-based

- **Prioritized Replay Buffer:** using TD error, give **priority** to “important” experiences.
- **Dueling DQN:** **Separate** Q-value estimation into state-value and advantage.
- **Double DQN:** Reduce overestimation using **a net for action selection** and another for Q-value estimation.
- **Noisy DQN:** Introduce **noise** into the network **weights** to encourage exploration.
- **Two-step DQN:** Use information from **two** time **steps ahead**, improving stability.

Reinforce: policy-based

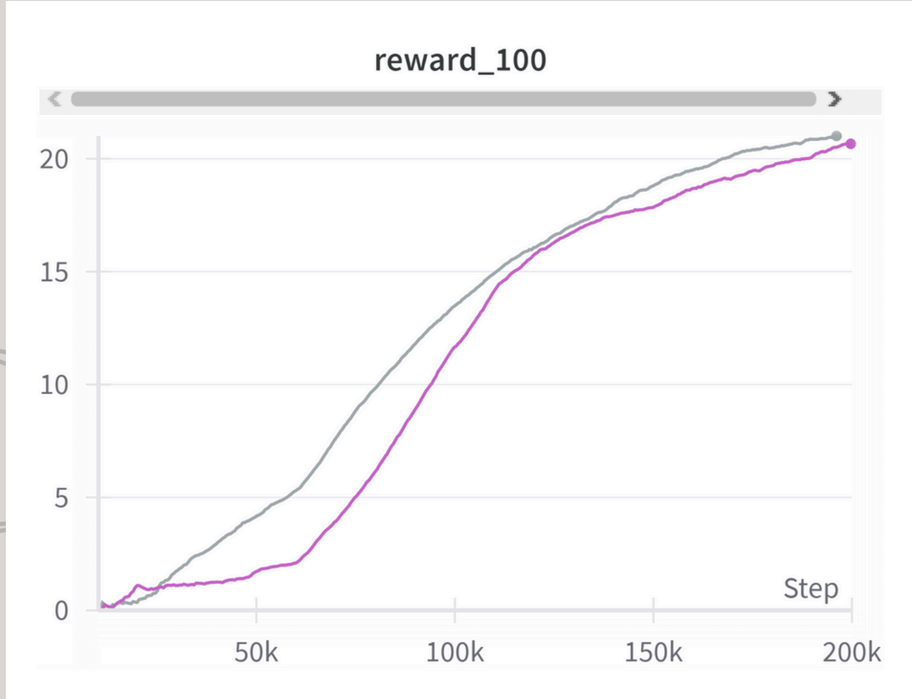
Quantitative results

DQN (+ extensions)



Quantitative results

DQN (+ extensions)

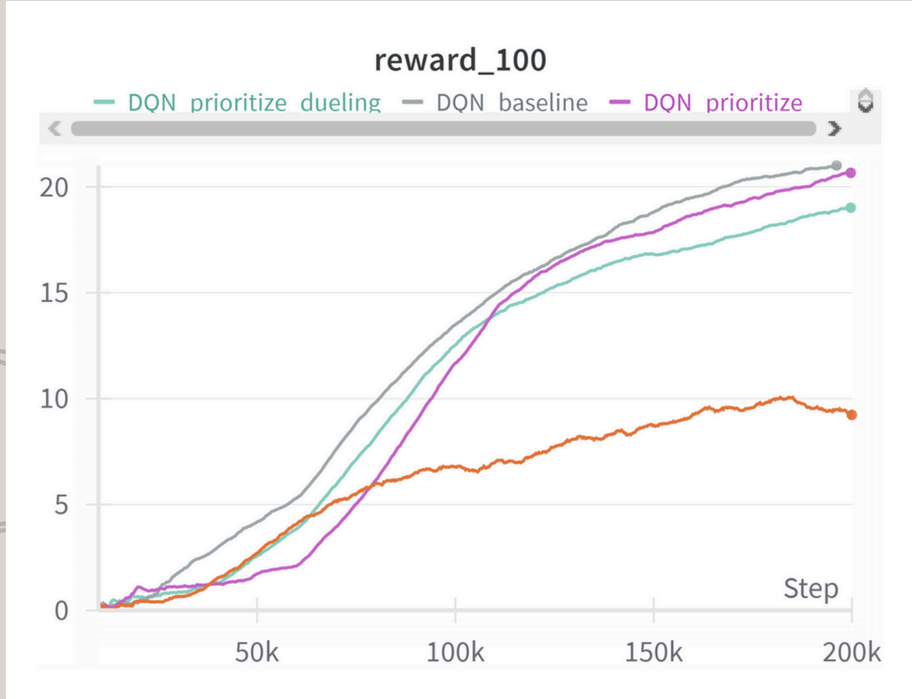


Quantitative results

DQN (+ extensions)

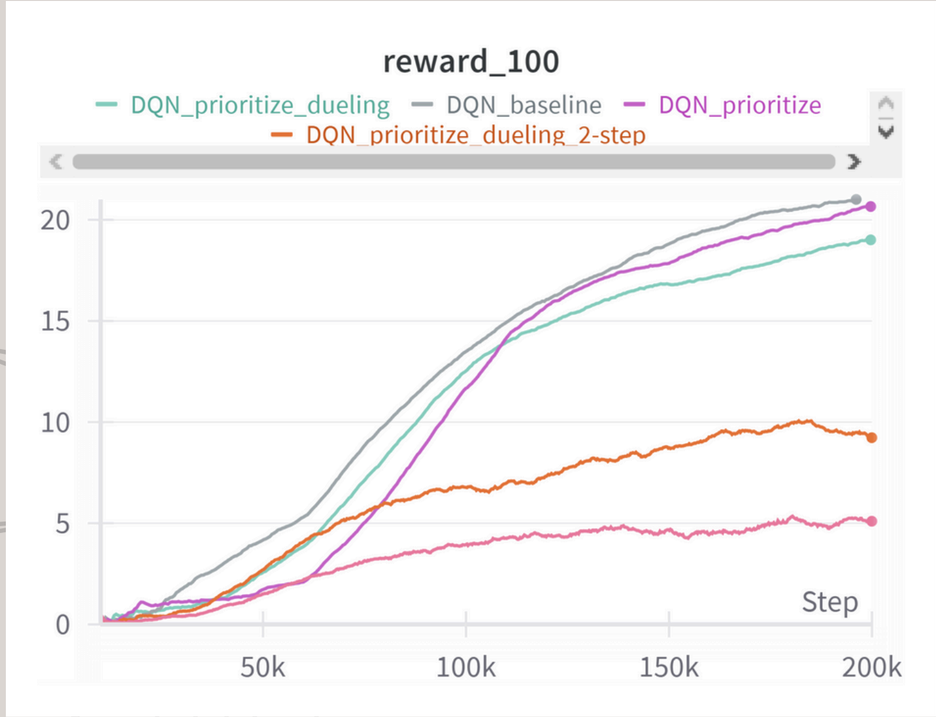
Quantitative results

DQN (+ extensions)



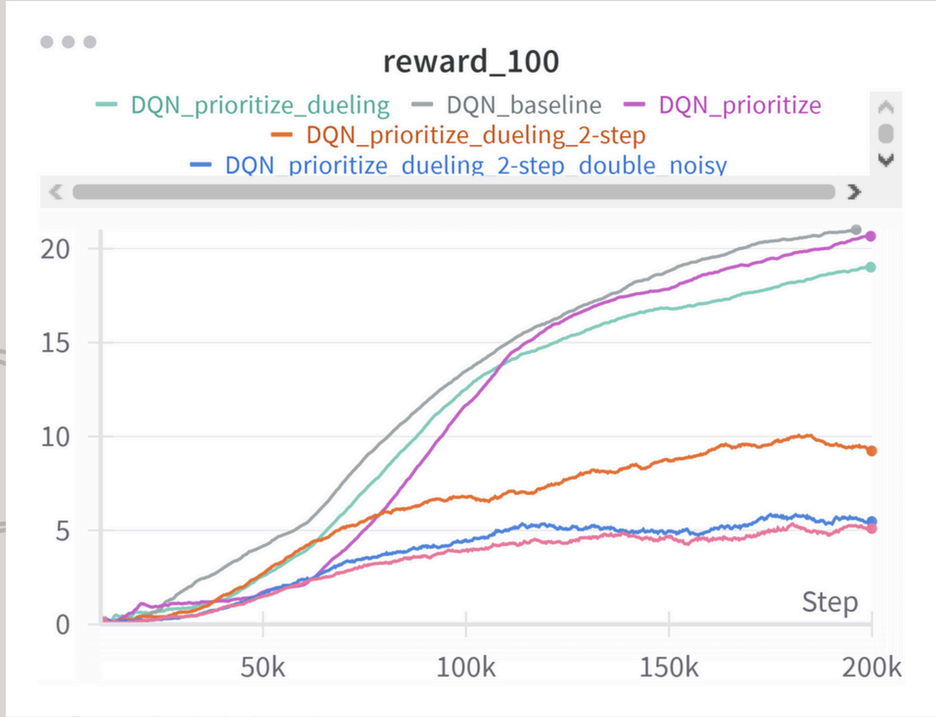
Quantitative results

DQN (+ extensions)



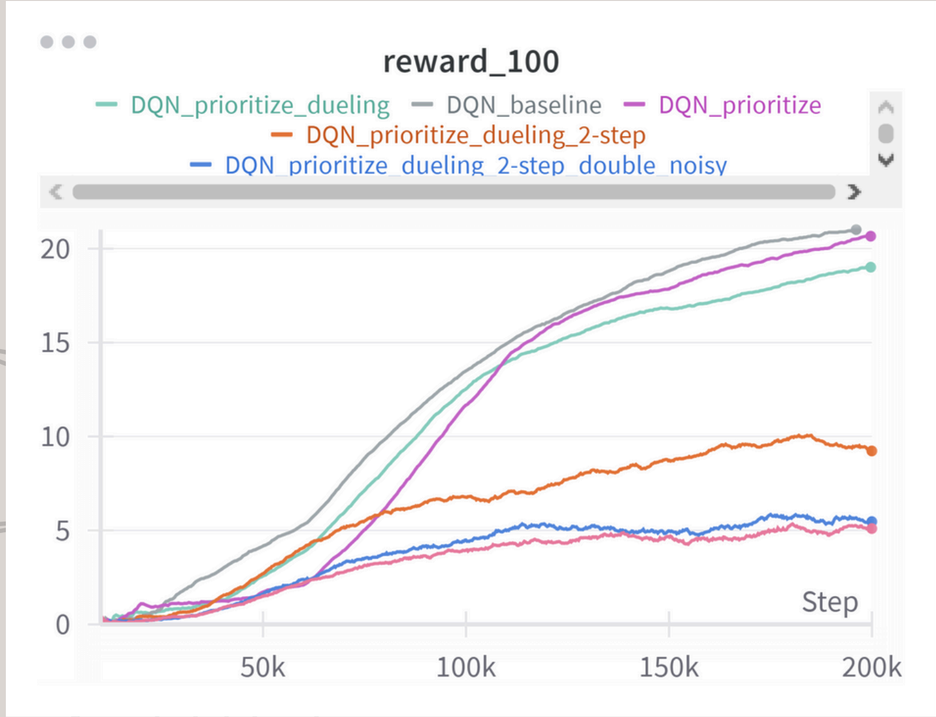
Quantitative results

DQN (+ extensions)



Quantitative results

DQN (+ extensions)

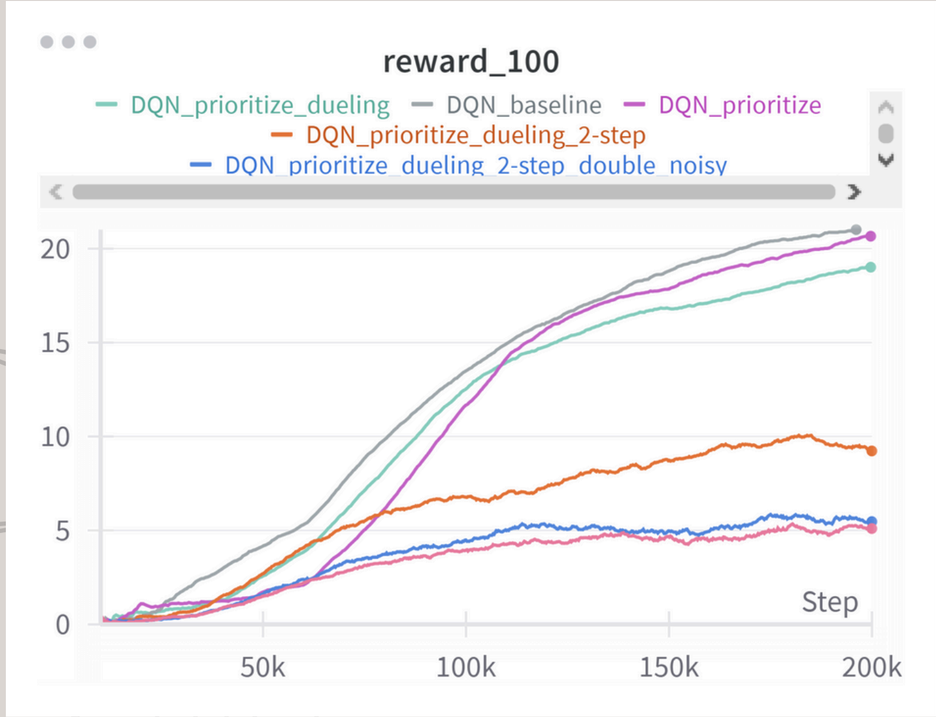


why extensions don’t improve?

- You are increassing the complexity
- RL algorithms are really sensitive to hyperparameters

Quantitative results

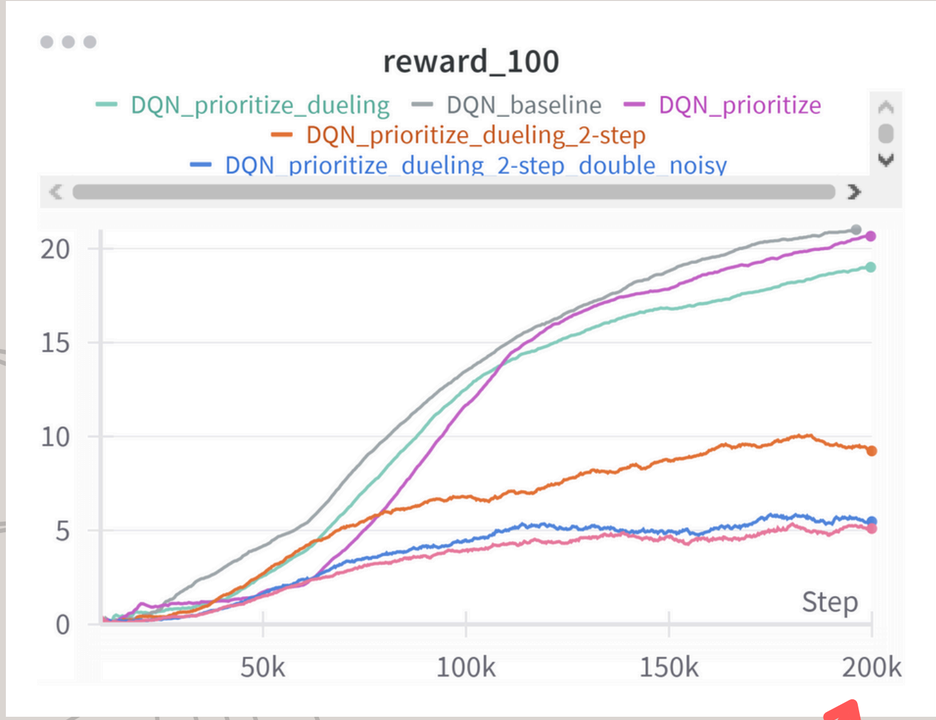
DQN (+ extensions)



Reinforce

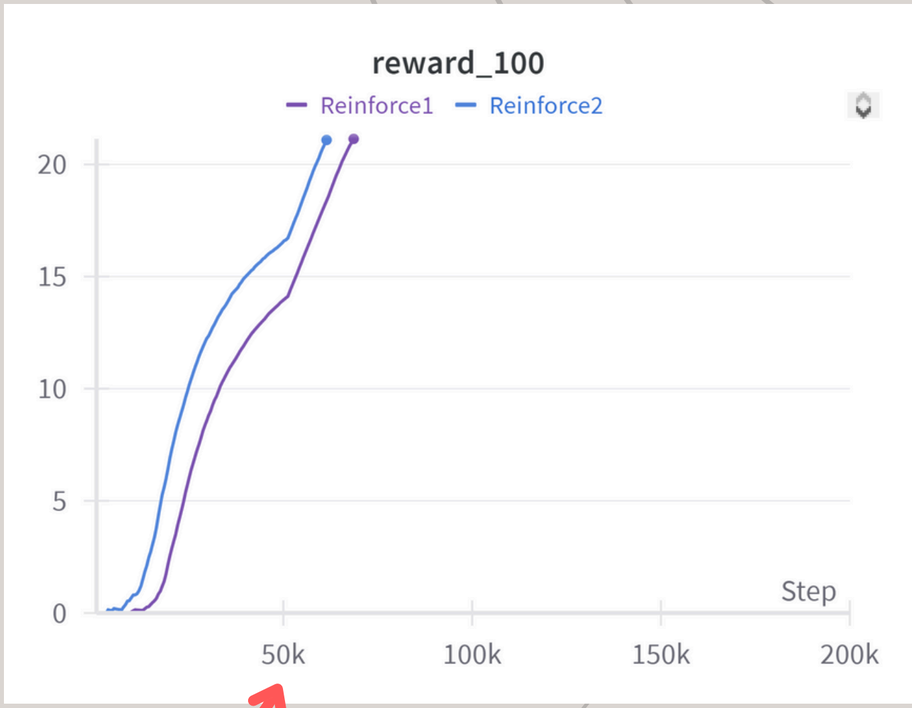
Quantitative results

DQN (+ extensions)



solved in 200k

Reinforce



solved in 50k!

Qualitative results

DQN

Reinforce

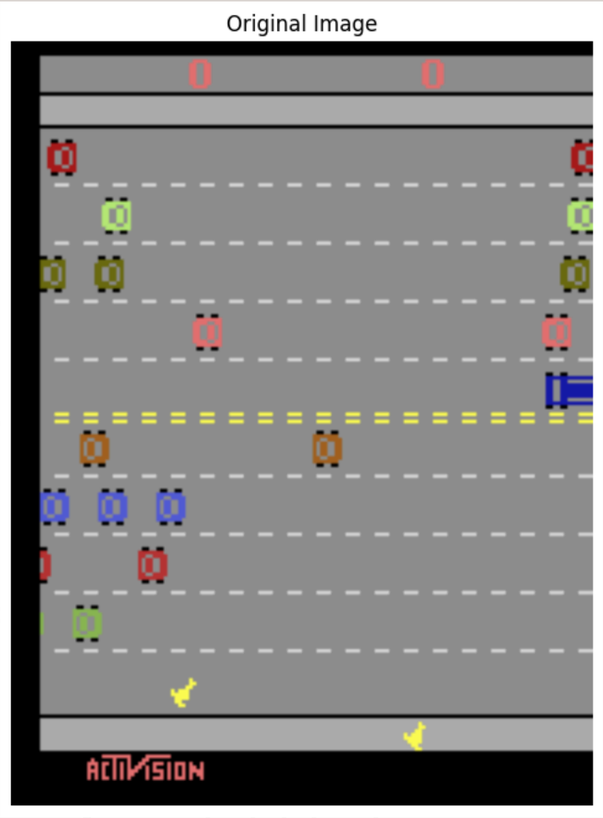
Other options we conteplated

What if we train a tabular method?

We would need to reduce our huge dimensional space (210 x 160 x 3) into a manageable lower dimensional space

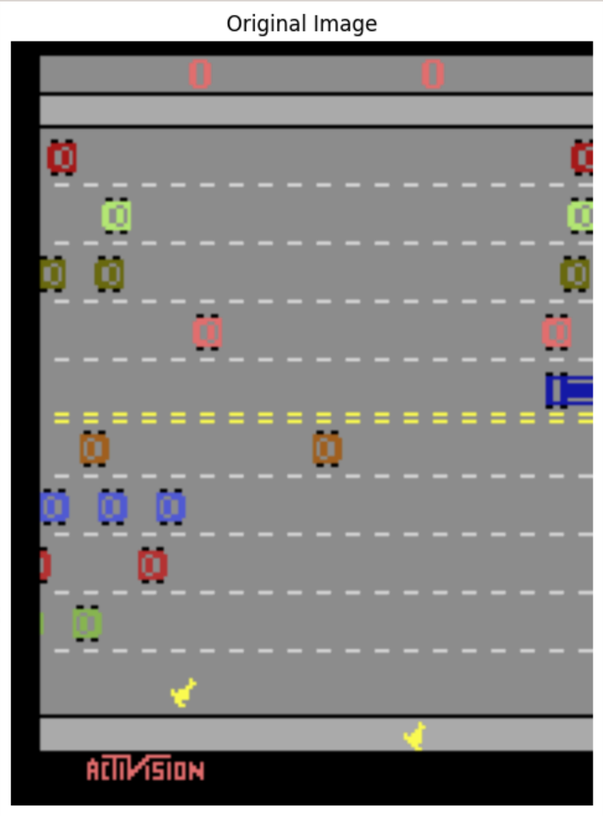
Discretitze somehow our environment

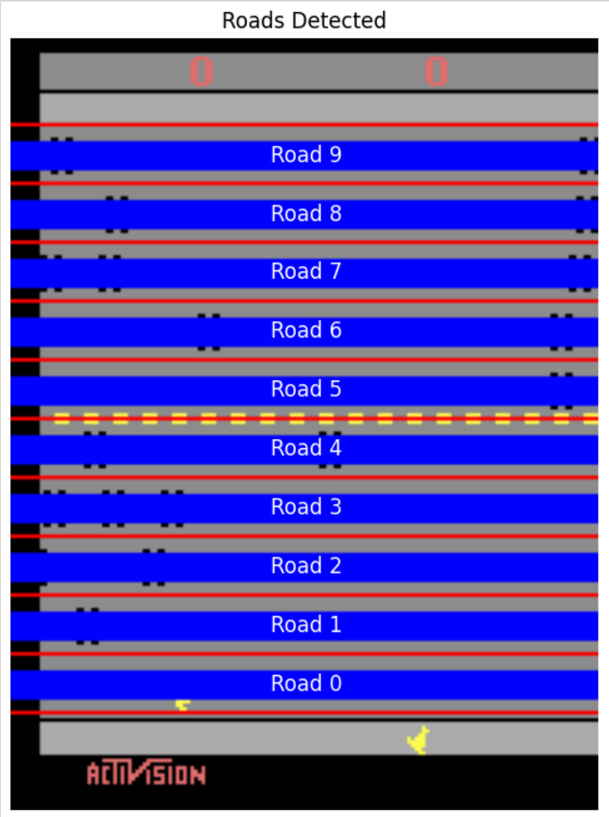
Discretitzicing an Atari Game:



Discretitzicing an Atari Game:

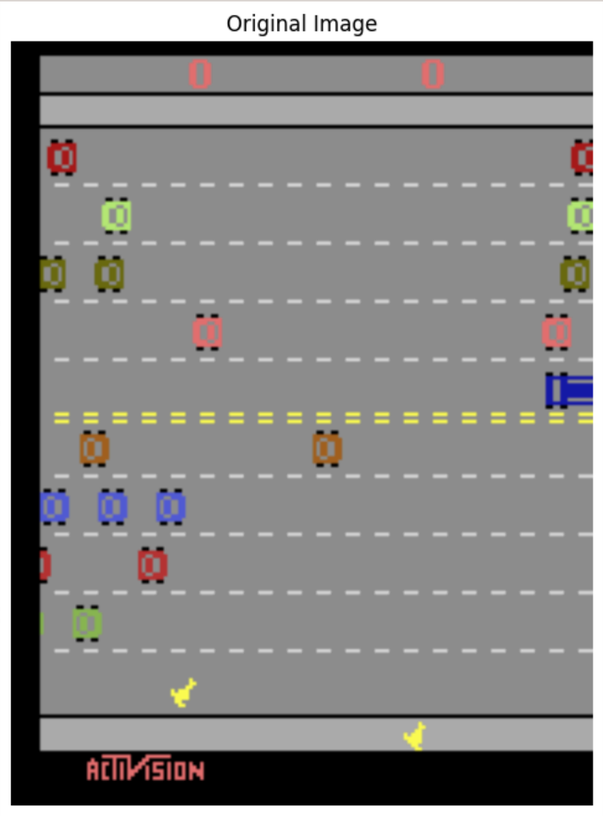
1. Detect Roads

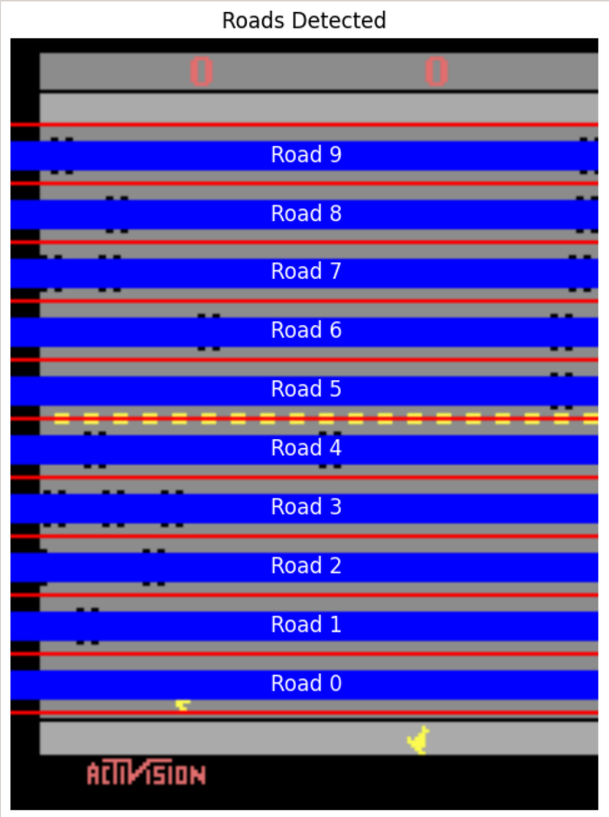


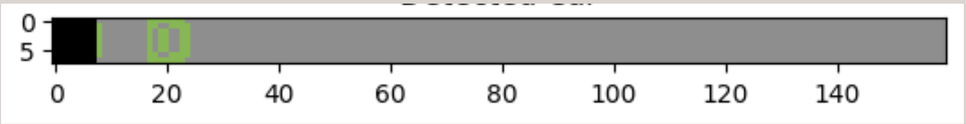


Discretitzicing an Atari Game:

1. Detect Roads

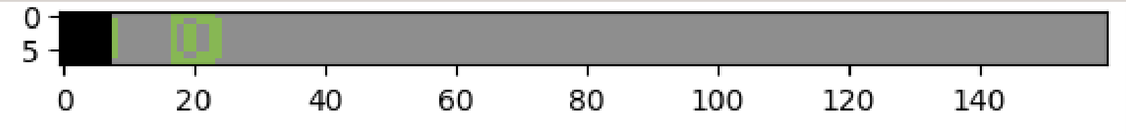






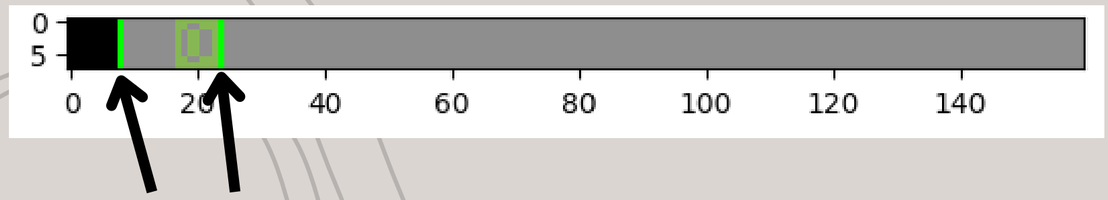
Discretitzicing an Atari Game:

2. **Detect cars**



On each road, we iterate over the columns looking for the ‘car pattern’

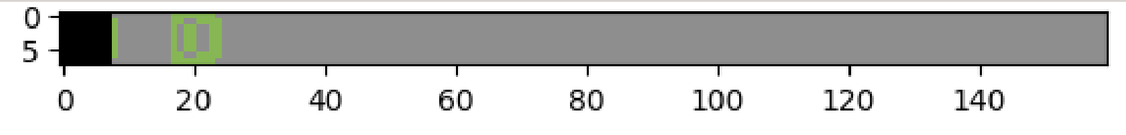
‘car pattern’
shape (8, 1)



Detected cars

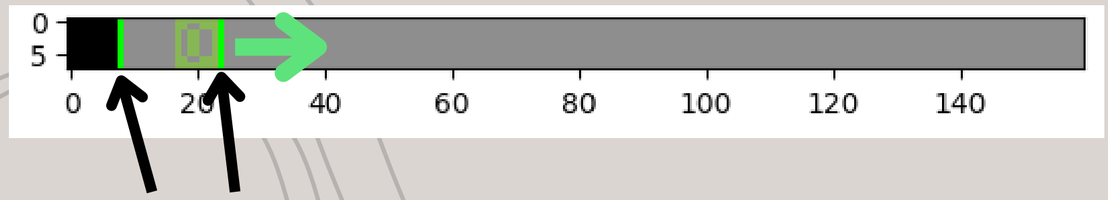
Discretitzicing an Atari Game:

2. **Detect cars**



On each road, we iterate over the columns looking for the ‘car pattern’

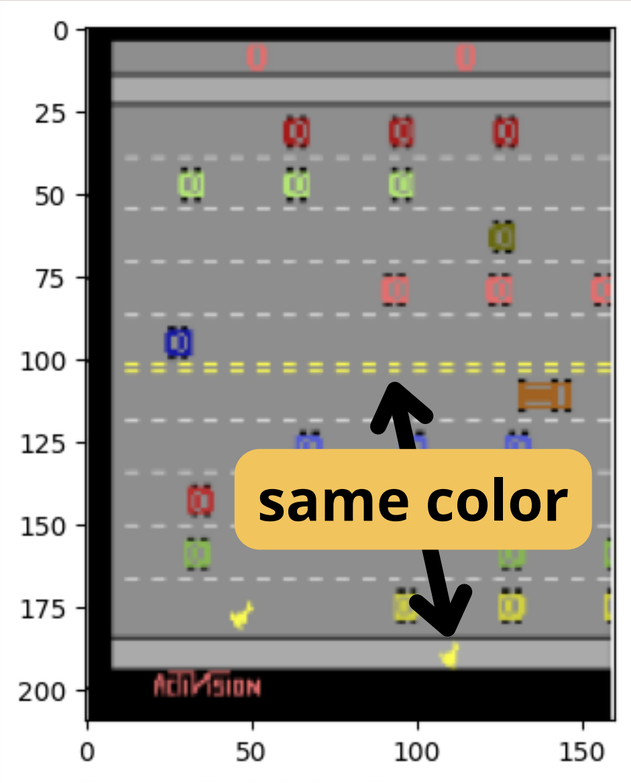
‘car pattern’
shape (8, 1)



Detected cars

Discretitzicing an Atari Game:

3. **Detect the bunny**



- We can directly search for its exact yellow color
- but, there dashed lines in the middle are also the same color
- Therefore, we search for the following pattern over the image.

‘bunny pattern’

Discretitzicing an Atari Game:

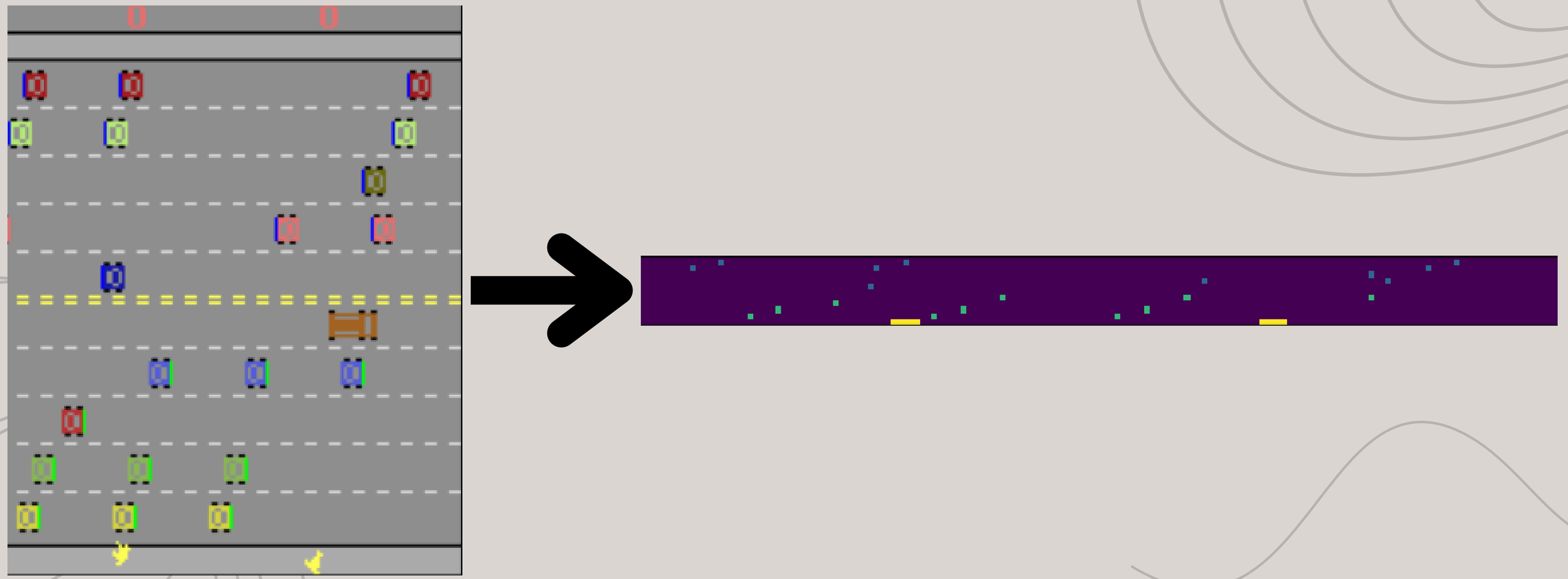
Okay, we know:

1. where the **roads** are
2. where the **cars** are and in which direction are going
3. where are we (the **bunny**)

We can simplify our complex game using this information

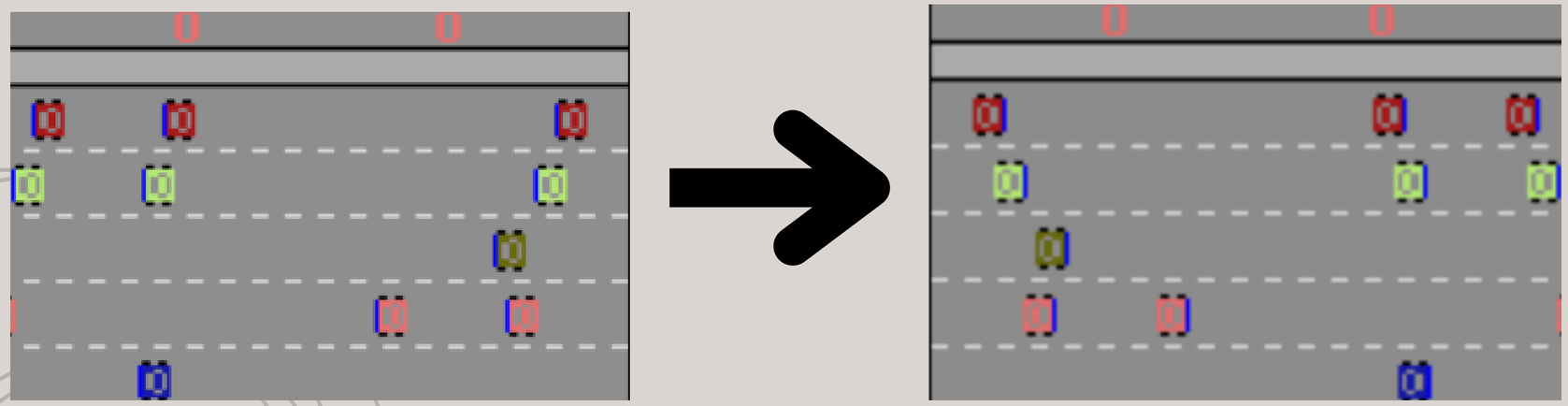
Discretitzicing an Atari Game:

We can simplify our complex game using roads, cars and the bunny



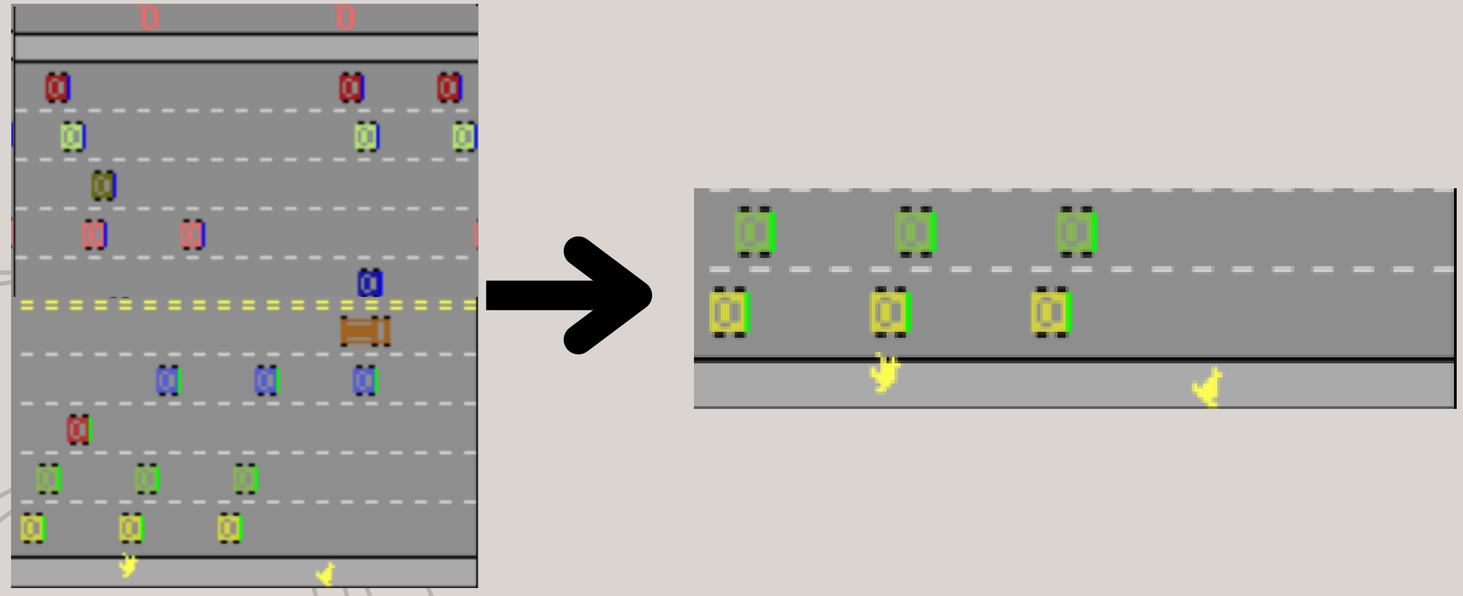
Discretitzicing an Atari Game:

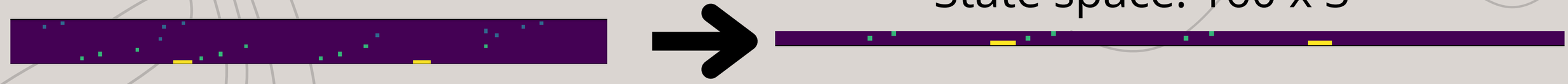
Next we reflect the cars going to the left so that all cars in the game “as the agent sees it” goes to the right.



Discretitzicing an Atari Game:

Finally, we use relative position by giving as state the pixels from the previous, current and next road.





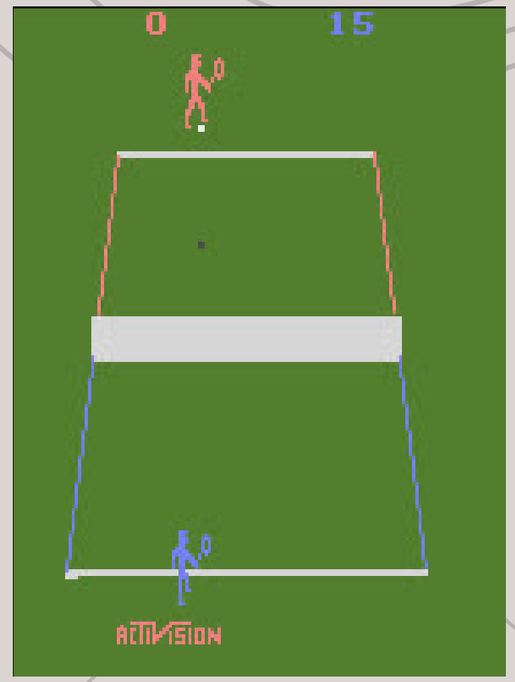
State space: 160 x 3

Complex Environment: Tennis

Description of the environment

Observation Space

Action Space	Discrete(18)
Observation Space	Box(0, 255, (210, 160, 3), uint8)

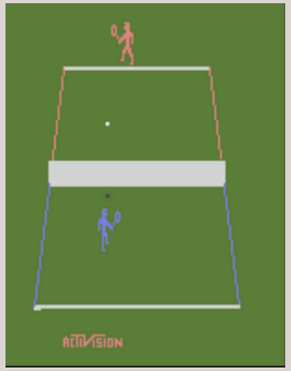


Available Actions

Value	Meaning	Value	Meaning	Value	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

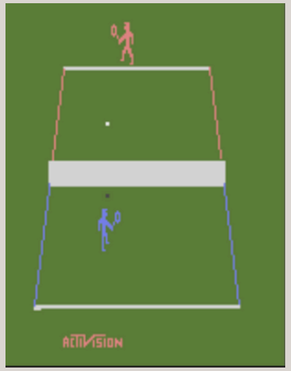
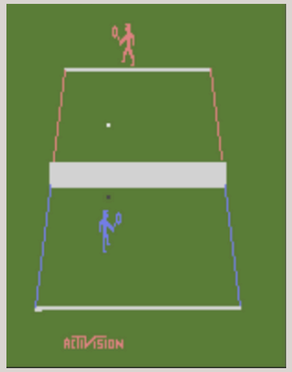
Pre-Processing

Original

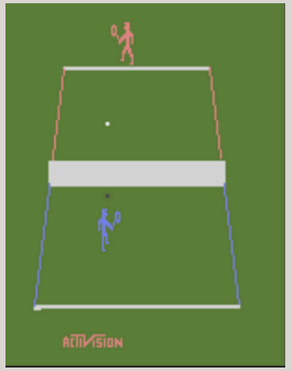
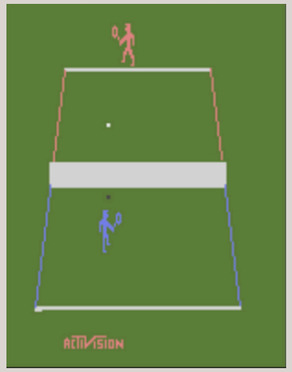
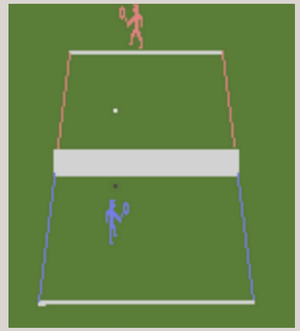


(210, 160, 3)

Pre-Processing

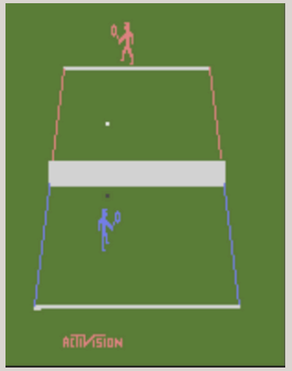
Original	MaxAndSkip
	
(210, 160, 3)	(210, 160, 3)

Pre-Processing

Original	MaxAndSkip	Resize	GrayScale
			
(210, 160, 3)	(210, 160, 3)	(84, 84, 3)	(84, 84, 1)

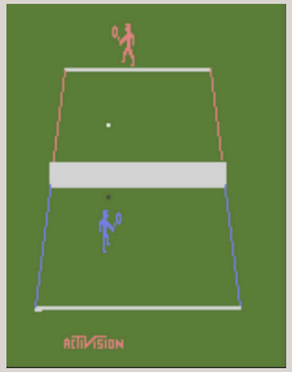
Pre-Processing

Original



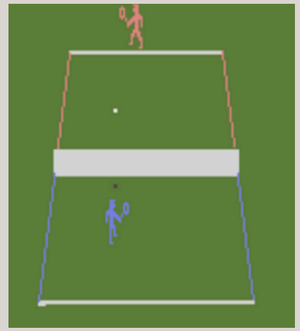
(210, 160, 3)

MaxAndSkip



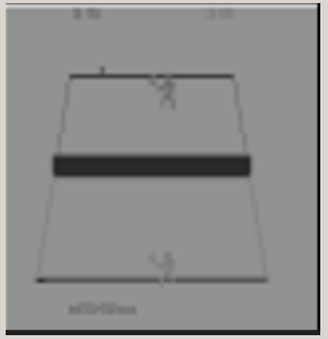
(210, 160, 3)

Resize



(84, 84, 3)

GrayScale



(84, 84, 1)

Reshape

(84, 84)

ImageToPyTorch

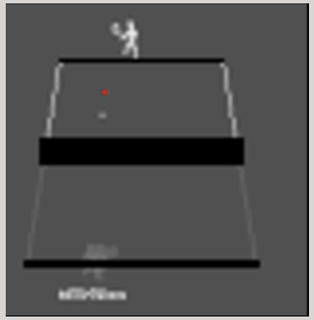
(1, 84, 84)

FrameStack

Frame 1

Frame 2

(4, 84, 84)



Frame 3

Frame 4

ScaledFloatFrame

Models trained

Proximal Policy Optimization (PPO)

- **Policy gradient**
- **On-Policy**

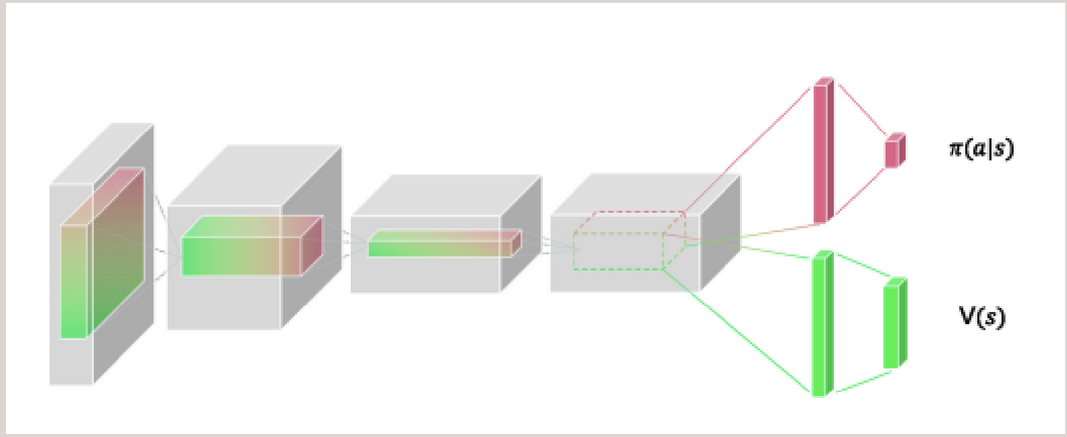
- **We know empirically that smaller policy updates during training are more likely to converge to an optimal solution.**

- **A too-big step in a policy update can result in falling “off the cliff” (getting a bad policy) and taking a long time or even having no possibility to recover.**

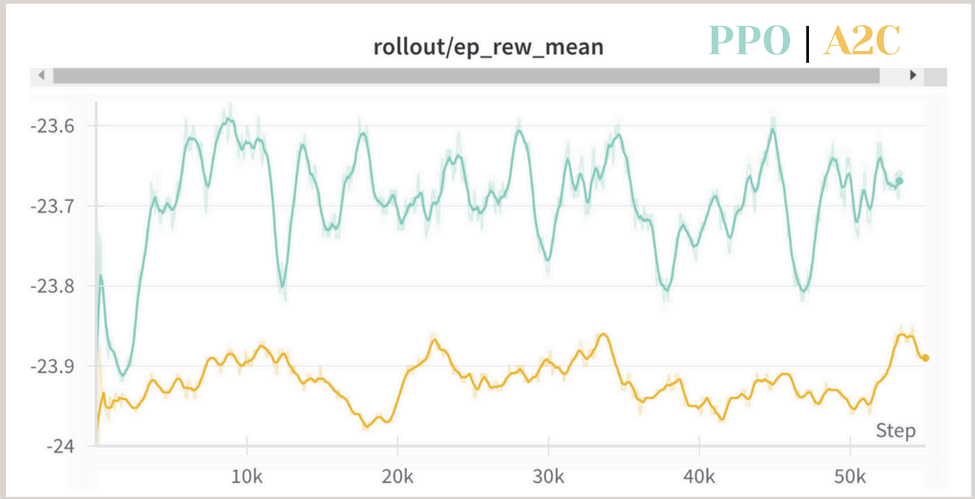
Models trained

Advantatge Actor Critic

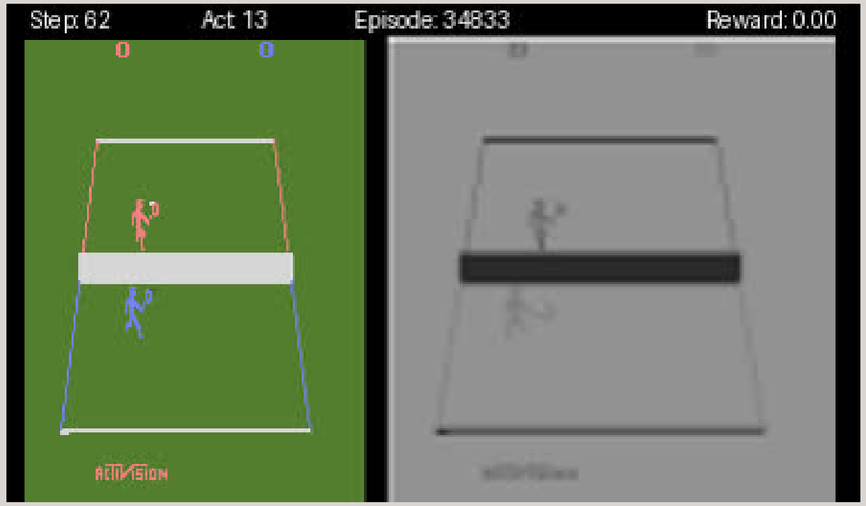
- **On-policy algorithm**
- **Actor-critic family, combining elements of both value-based and policy-based methods**



Evaluation



PPO



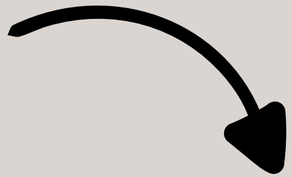
A2C

Why is this game specially complex?

- **Sparse Rewards:** The rewards are given at the end of the points
- High **number of actions**: 18 actions
- Visually complex scene
- In the middle of the game you change sides with the opponent.

Observations

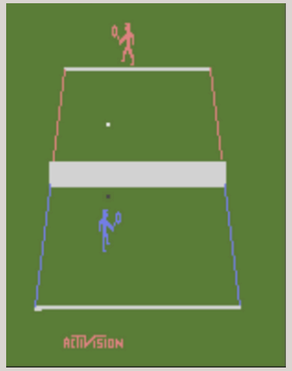
Value	Meaning	Value	Meaning	Value	Meaning
0	NOOP	1	FIRE	2	UP
3	RIGHT	4	LEFT	5	DOWN
6	UPRIGHT	7	UPLEFT	8	DOWNRIGHT
9	DOWNLEFT	10	UPFIRE	11	RIGHTFIRE
12	LEFTFIRE	13	DOWNFIRE	14	UPRIGHTFIRE
15	UPLEFTFIRE	16	DOWNRIGHTFIRE	17	DOWNLEFTFIRE



Value	Meaning	Value	Meaning	Value	Meaning
0	NOOP	1	FIRE	2	UP
3	RIGHT	4	LEFT	5	DOWN

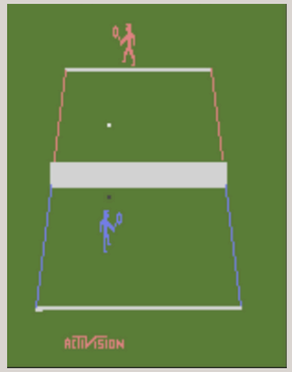
New Pre-Processing

Original



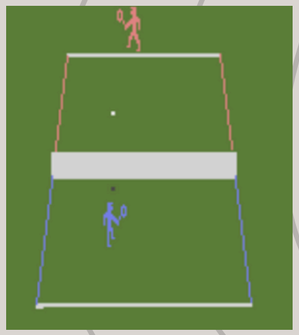
(210, 160, 3)

MaxAndSkip



(210, 160, 3)

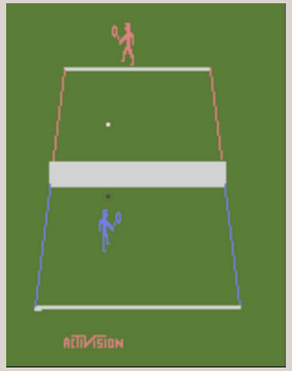
Resize



(84, 84, 3)

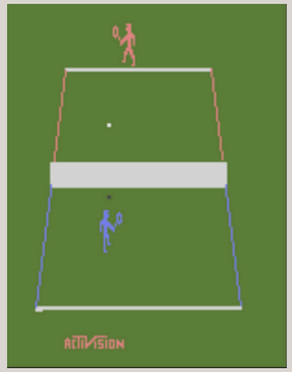
New Pre-Processing

Original



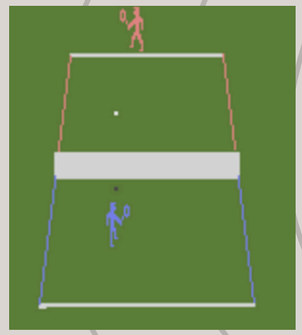
(210, 160, 3)

MaxAndSkip



(210, 160, 3)

Resize

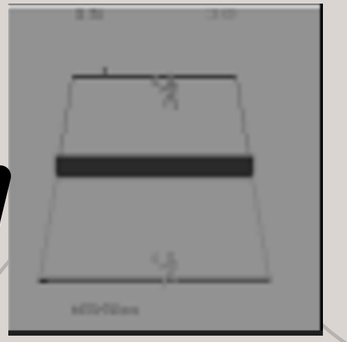


(84, 84, 3)

RedChannel & B.Position

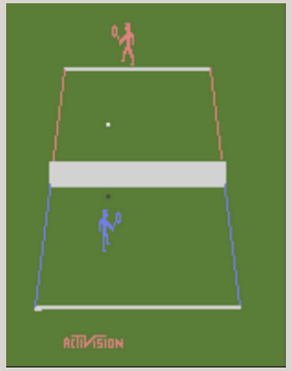
(84, 84, 1)

Before:
GrayScale



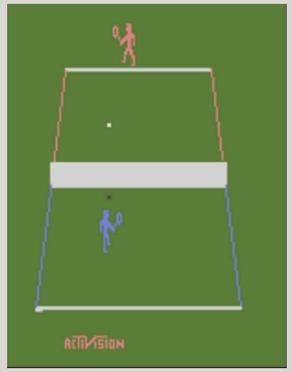
New Pre-Processing

Original



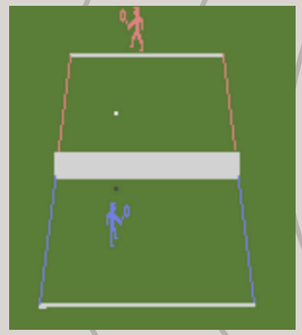
(210, 160, 3)

MaxAndSkip



(210, 160, 3)

Resize



(84, 84, 3)

RedChannel & B.Position

(84, 84, 1)

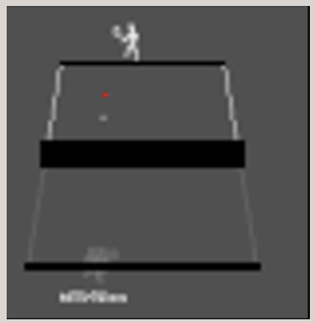
Reshape

(84, 84)

ImageToPyTorch

(1, 84, 84)

FrameStack



Frame 1

Frame 2

(4, 84, 84)

Frame 3

Frame 4

ScaledFloatFrame

TennisWraper

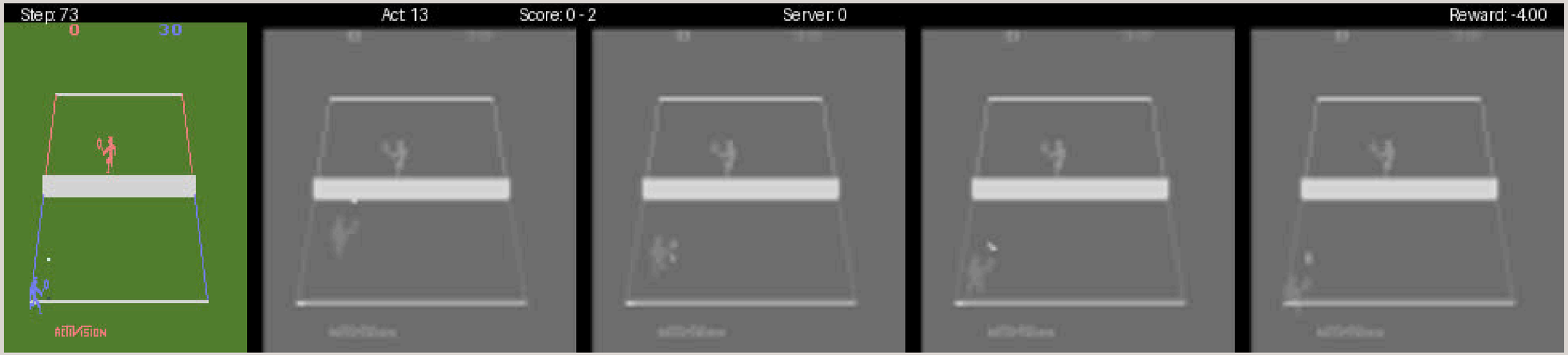
Responsible of storing:

- Current Score
- Server

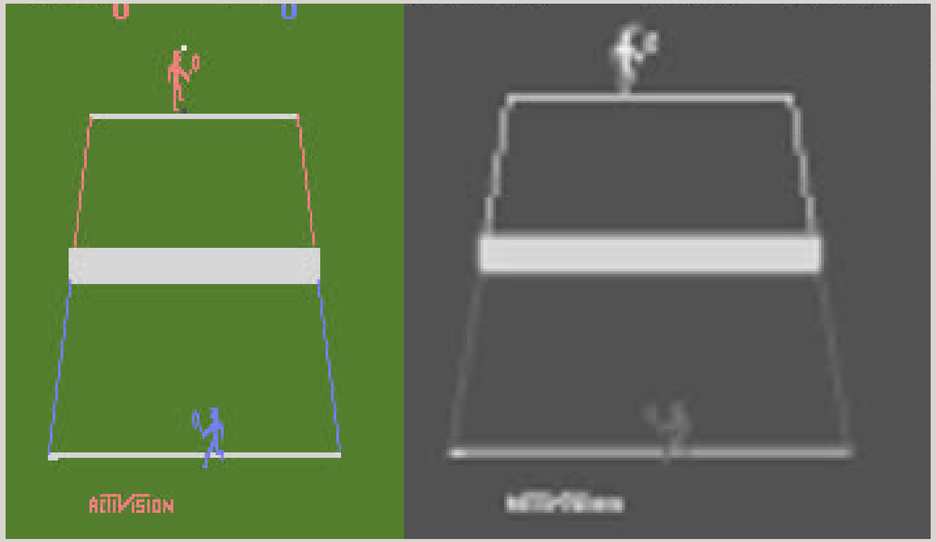
RewardWraper

Responsible of:

- Penalize Based on the frames seen
- End the game, as soon the point finishes



Suboptimal Solutions



If you dont serve, the reward is neither positive nor **negative**

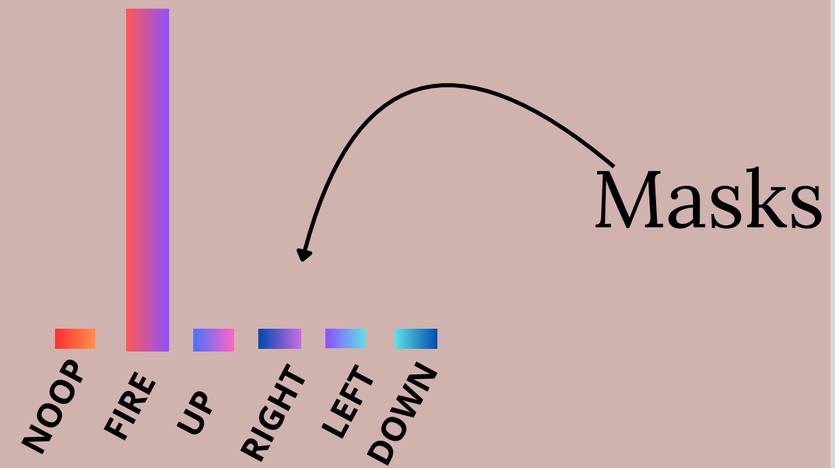
Prioritize certain actions over others during training.

Models Trained

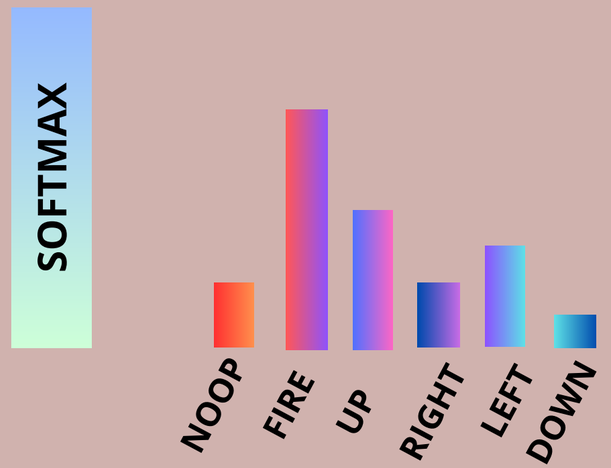
- Maskable PPO
- Maskable A2C

Mask illegal actions

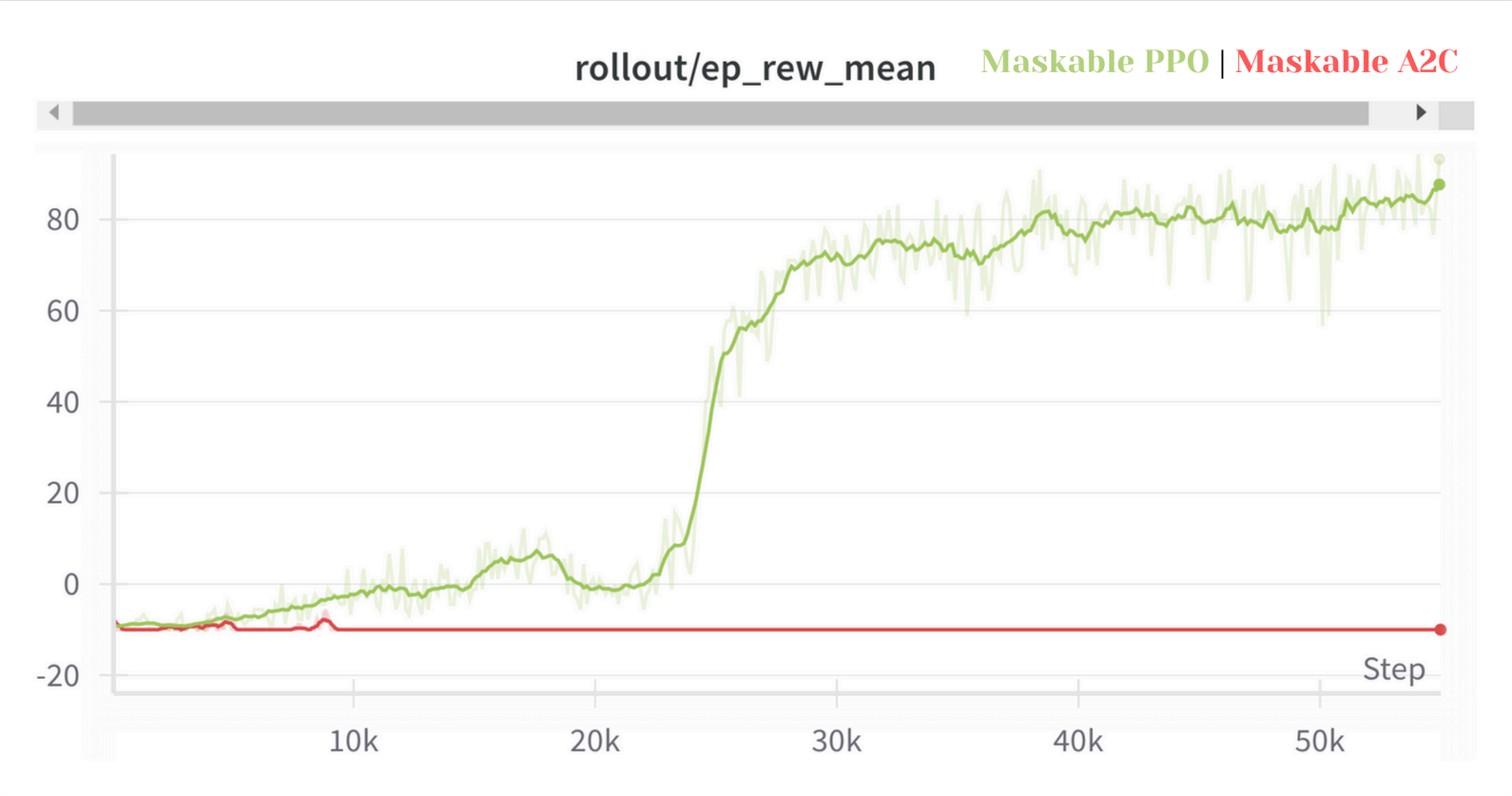
if Server = 0 and frames < 10



Server = 1

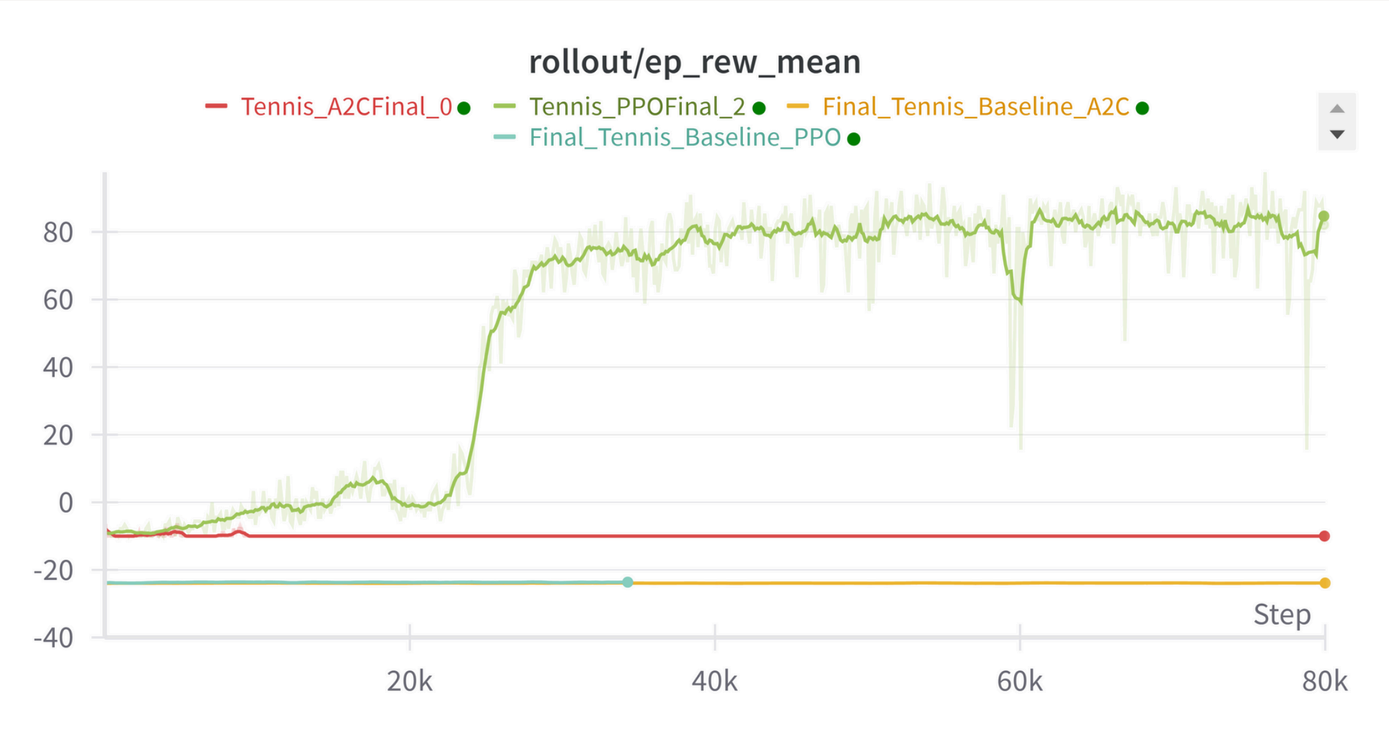


Quantitative results



- **Maskable PPO** manages to learn, getting an avergage reward of 80
- **Maskable A2C** gets stuck on a local minima and never manages to learn anything meaningful

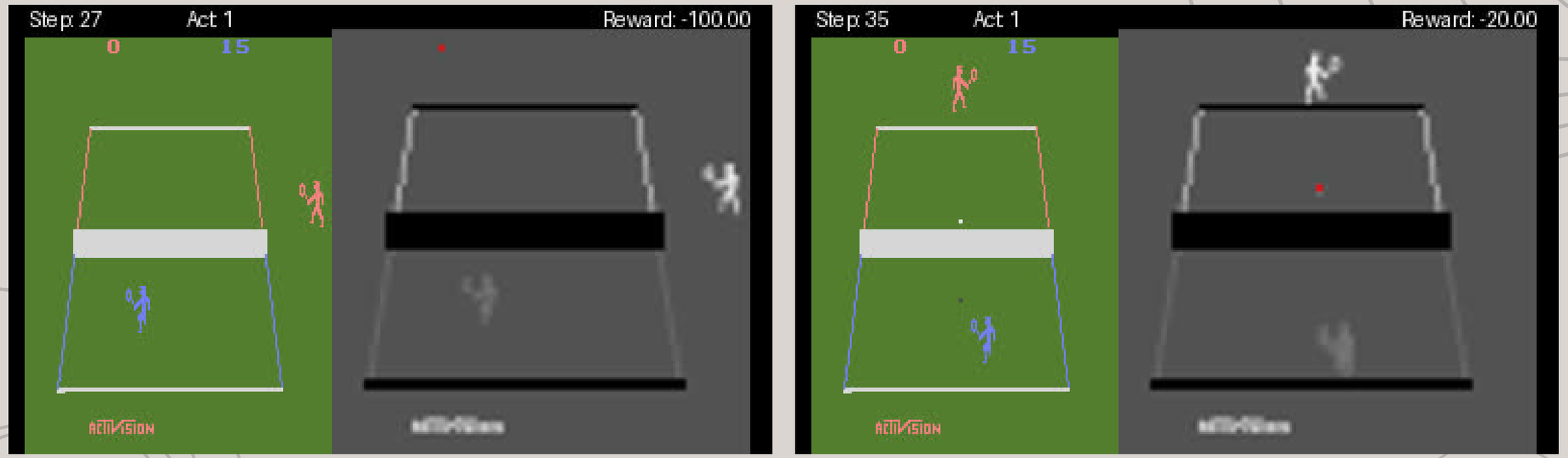
Quantitative results



- **Baseline models:** Find as best solution not serve the ball.

- **With our modifications:** we manage to get a model that always serves and using PPO model can consistently get points. However, is not good enough to win the whole match

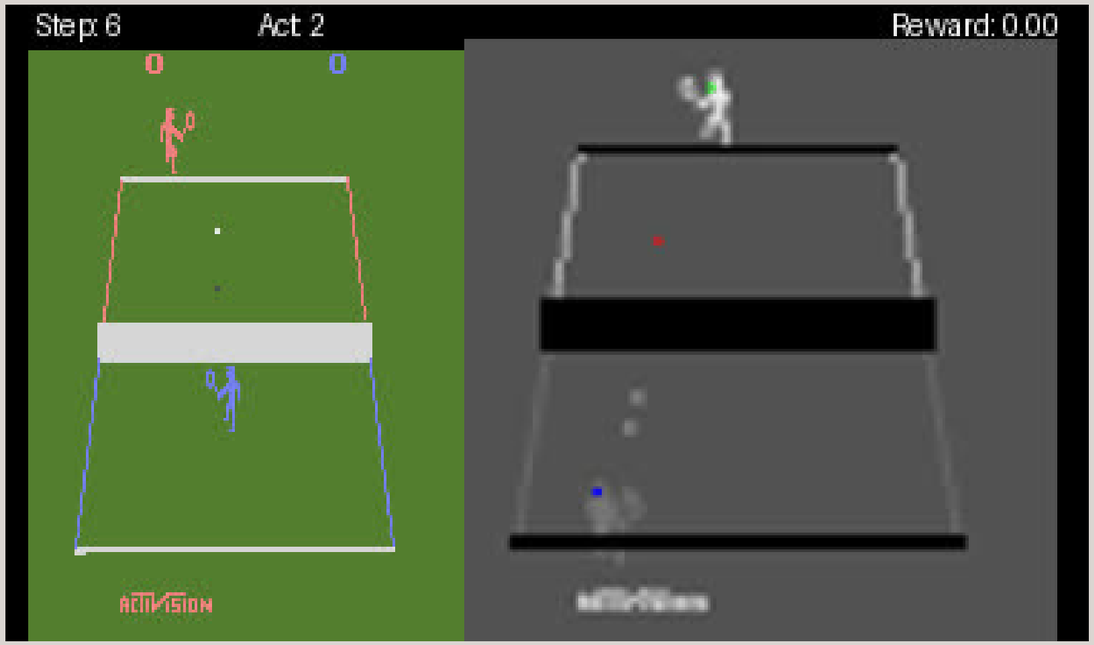
Qualitative results: Maskable PPO

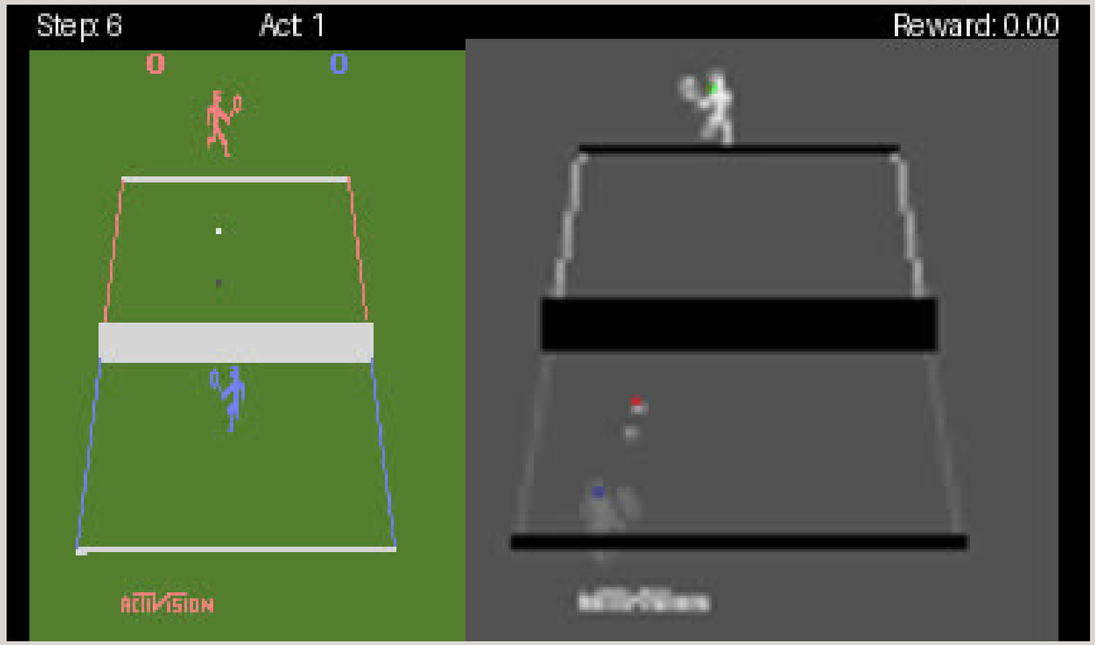


Qualitative results: Extended

Information: Positions of all the elements in the game

Model: Simple MLP



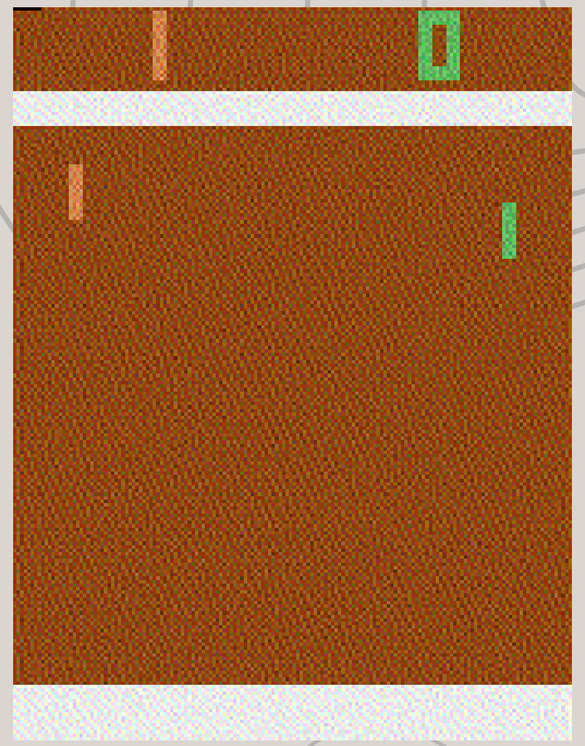


Pong World Tournament

Description of the env

Observation Space	(210, 160, 3)

Num	Action
0	NOOP
1	FIRE
2	RIGHT
3	LEFT
4	RIGHTFIRE
5	LEFTFIRE



Single Agent Approach

Single Agent Approach

Pre-processing: default make_atari_env from gymnasium

Single Agent Approach

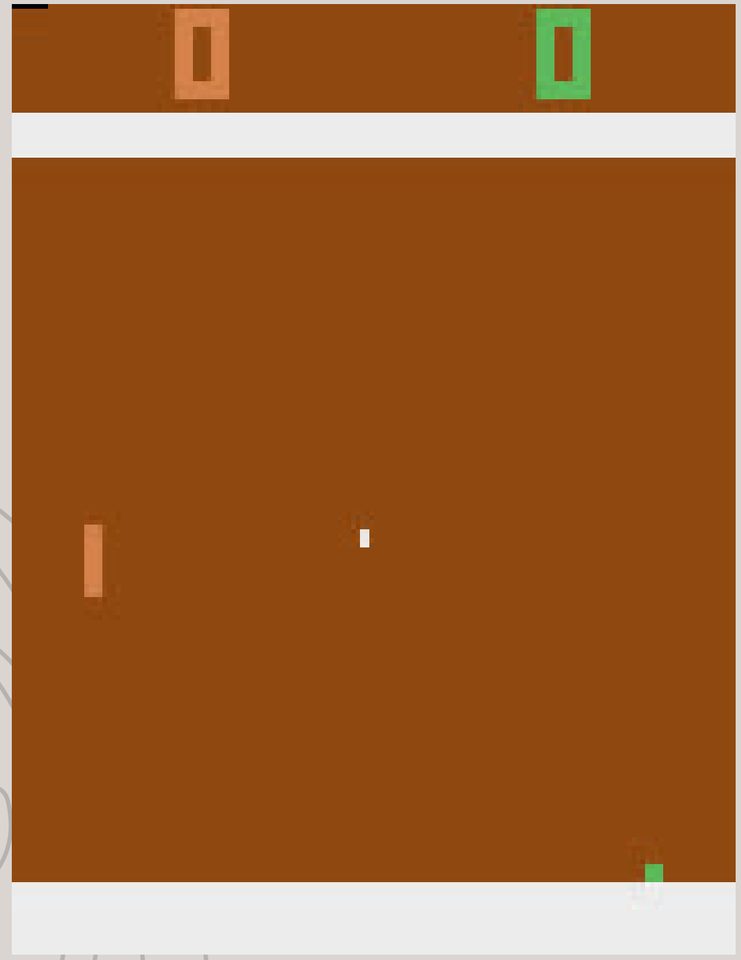
Pre-processing: default make_atari_env from gymnasium

Models trained:

- Reinforce
  - 440 episodes
  - Our code
- PPO
  - 600.000 timesteps
  - Stable baselines 3

Single Agent Approach

Results: Reinforce



the agent learns to go to the corner and not move :(

Single Agent Approach

Results: PPO, our best model.

100% winrate :D

Multi Agent Approach

Multi Agent Approach

Pre-processing: wrappers given for the tournament

Multi Agent Approach

Pre-processing: wrappers given for the tournament

Goal: Use **PettingZoo** directly and train 2 agents simultaneously by playing one agains each other.

Multi Agent Approach

Pre-processing: wrappers given for the tournament

Goal: Use **PettingZoo** directly and train 2 agents simultaneously by playing one against each other.

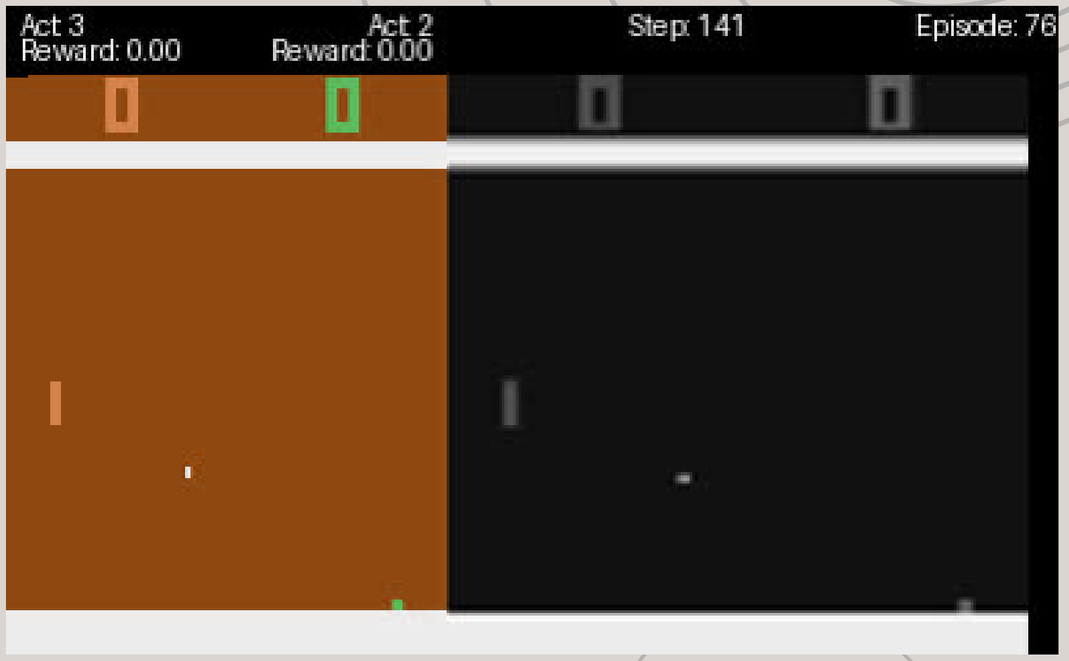
Models trained:

- **Reinforce**
- **PPO**

Multi Agent Approach

Reinforce

PPO



Problem! Suboptimal solutions!

Multi Agent Approach

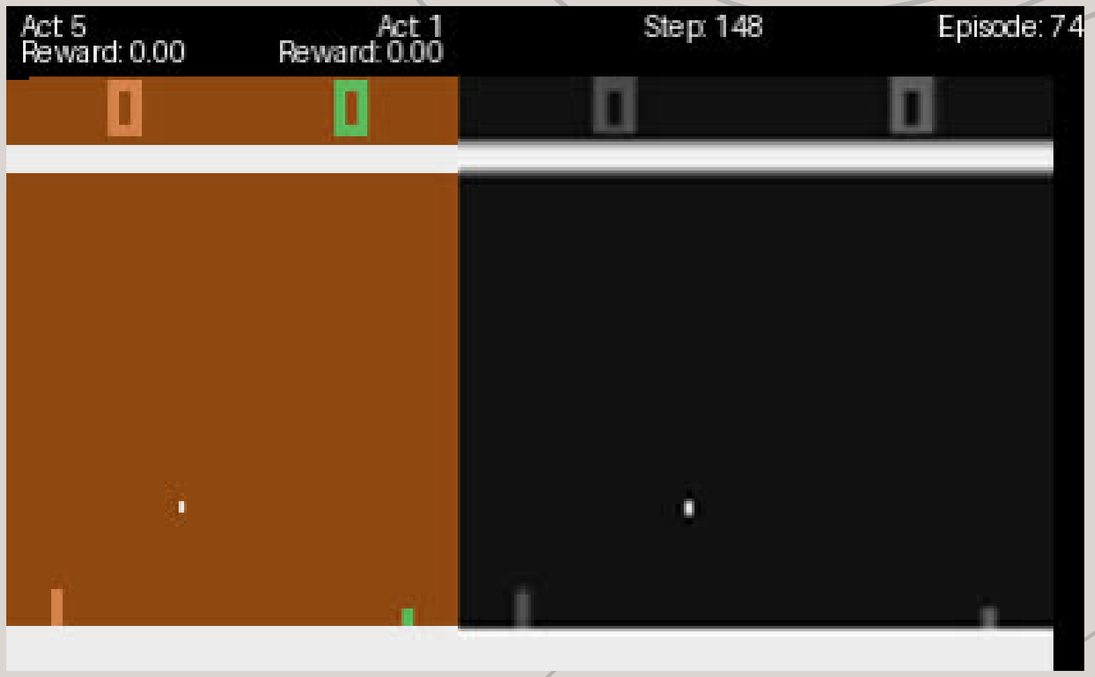
Idea: Train agains our already trained agent and either:

- Freeze the trained agent
- Finetune the trained agent

Multi Agent Approach

Reinforce training the left agent against our frozen trained PPO agent

PPO training the left agent and finetunning our right trained PPO agent



Still not good enough :(

Multi Agent Approach

Idea: Train agains our already trained agent and either:

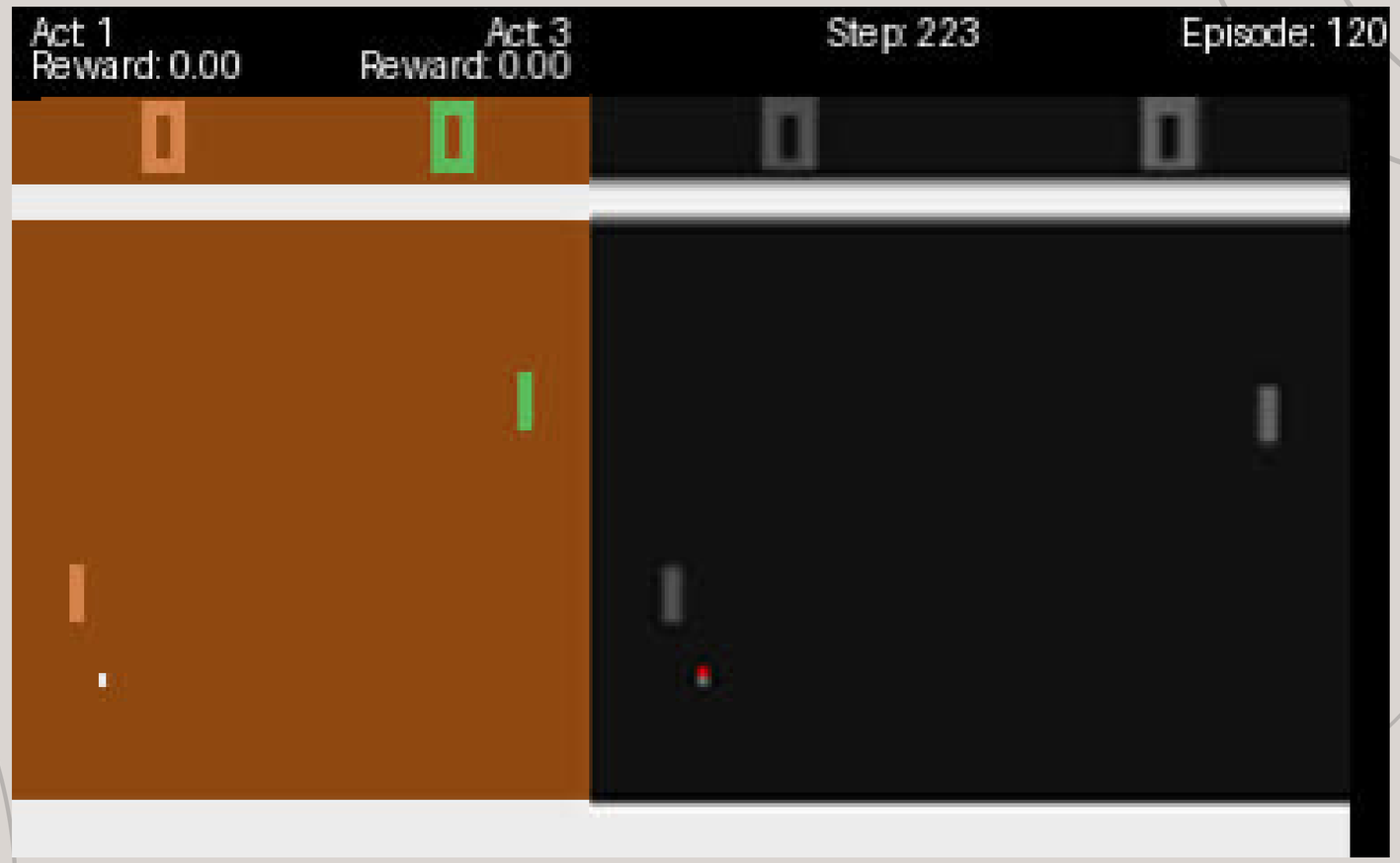
- Freeze the trained agent
- Finetune the trained agent

Reduce the action space and **like in tennis, track the ball position!**

Multi Agent Approach

Better! This is our best left model.

PPO



Conclusions

3 environments:

- **Freeways:**
  - Solved with DQN (+ extensions) and Reinforce.
  - Reinforce outperforms DQN by a lot. Solves the env much faster.
- **Tennis:**
  - Lots of tricks to try to make it work, however, it is not solved.
  -
- **Pong:**
  - Single agent approach solved using PPO. Reinforce doesn’t get a good solution.
  - We managed to make the multi agent approach work, however, doesn’t solve the environment.

Thank you!


4 0
ACTIVISION
Step: 81 Act 1 Episode: 600000 Eps: 0.00 Reward: 0.00
Act: 1
Reward: 0.00
Act: 3
Reward: 0.00
Step: 223
Episode: 120
Step: 6 Act 1 Reward: 0.00
0 0
ACTIVISION