# Deep Learning: Project presentation

## Deep Reinforcement Learning with JAVAgario

DEBES Baptiste

NELISSEN Louis

WEYDERS Pierre-Francois

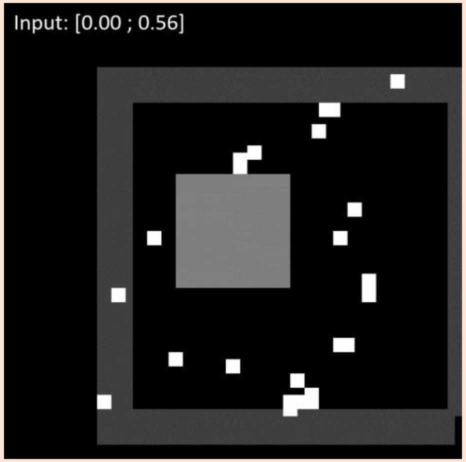
August 2020

# The Game: Agar.io

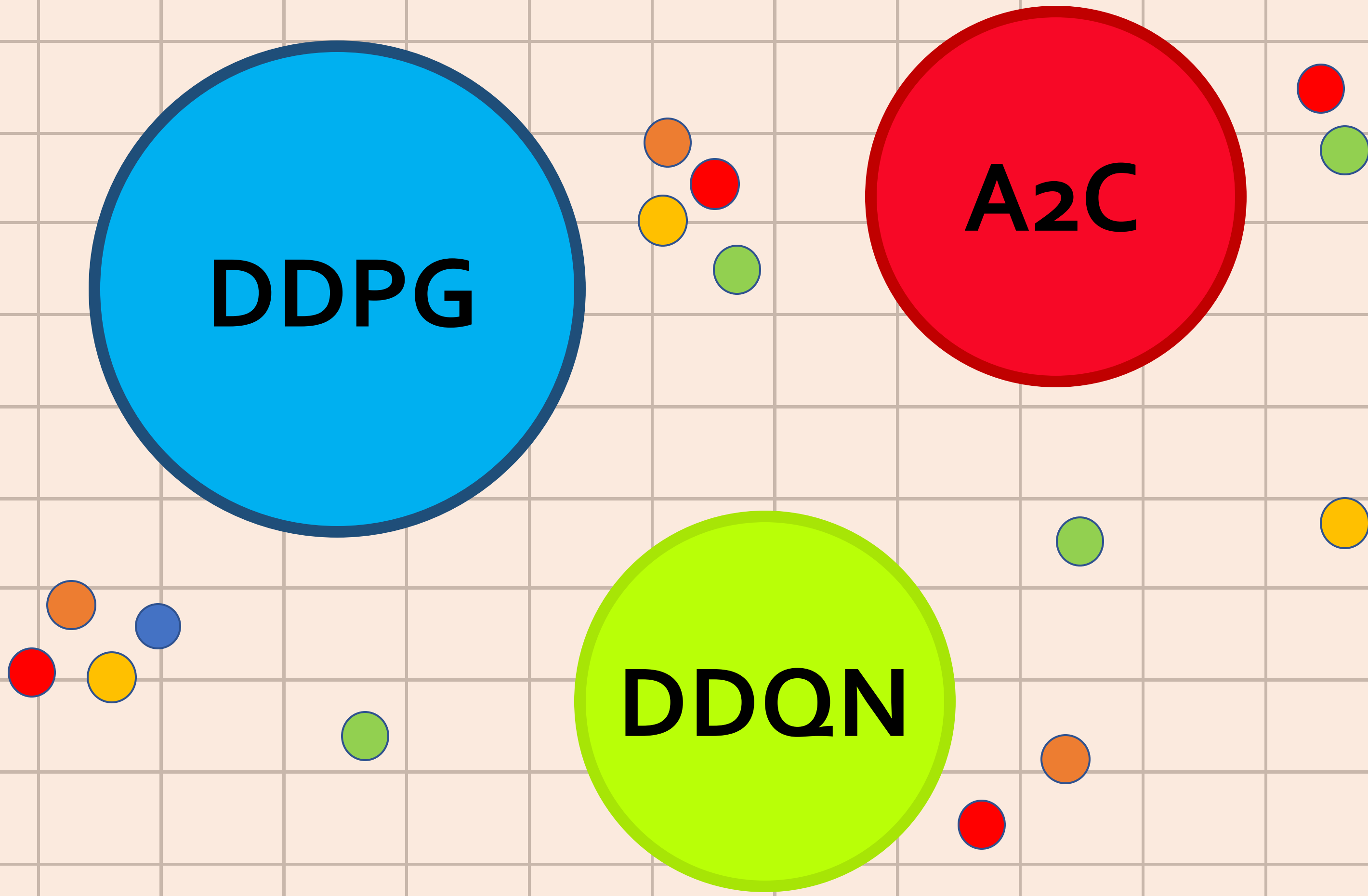
- Popular Massively Multiplayer browser game
- Player is a cell that eats pellets
- Constantly shrinks, but grows when eating pellets
- Controlled with mouse

# The Game: JAVAgar.io

- Coded in JAVA
- Greyscale image
- Ability to launch our own servers
- Simplified rules
  - One agent
  - One box (small)
  - 20 pellets
- Objective: get all the pellets as fast as possible without dying



# The Networks



# The Theory: Q-Learning

- Branch of reinforcment learning
- Model-free
- Value based learning
  - Q → (**s**tate, **a**ction)
- $Q$-learning aims to approximate the optimal $Q^*$
- Deep Q-learning: use NN as approximators

# The Theory: Policy Gradient

- Branch of reinforcment learning
- Model-free
- Seek an approximator $\pi_\theta(a|s)$ of the optimal policy $\pi^*(a|s)$

# The Theory: The RL family

**Value based** DDQN

- No policy (implicit)
- Value function

**Policy based**

- Policy
- No value function

**Actor critic** DDPG A2C

- Policy
- Value function

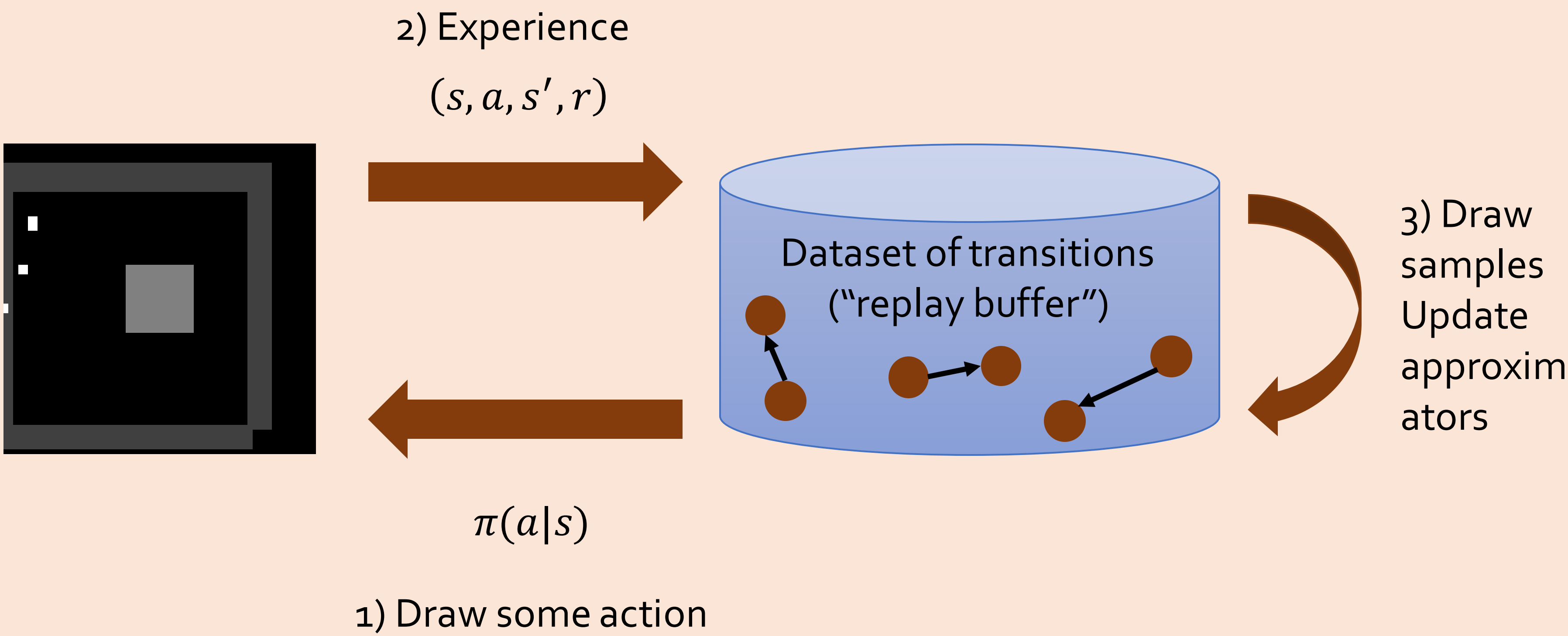
**Model based**

- Transition and reward model

**Model free**

- No transition and no reward model (implicit)

# The Theory: The general algorithm



# The Theory: The losses

## A2C: Advantage Actor Critic

- $L^{PG}(\theta) = E\left[ -\log\left(\pi_\theta(a_n|s_n)A(s_n, a_n)\right) \right]$

---

## DDPG: Double DPG

- $L^{DDPG}(\theta) = -E\left[Q_\tau(s, a)\,|_{s=s_n, a_n=\pi_\theta(s_n)}\right]$

Caveat in the report

---

## DDQN: Double Deep Q Network

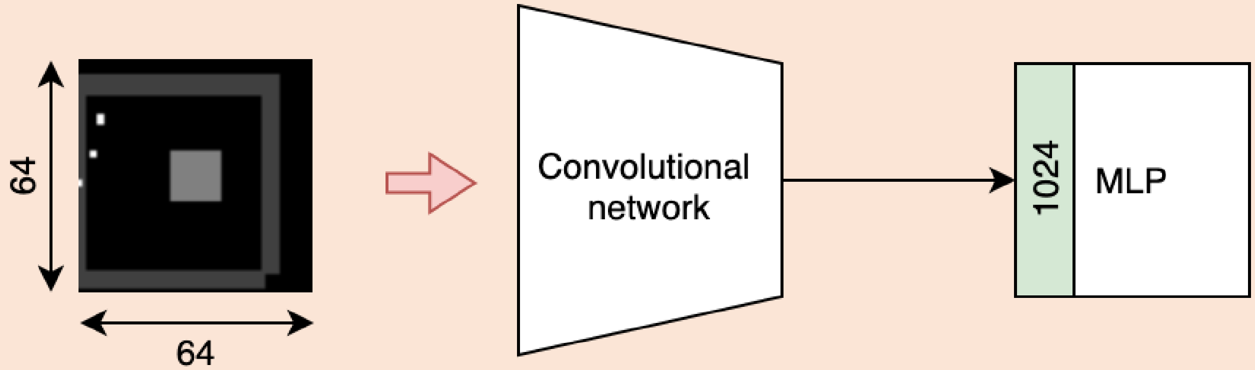
- $L^{TD}(\theta) = \frac{1}{2}\left[Q_\theta(s, a) - r - \gamma \max_{\{a'\}} Q_{\overline{\theta}}(s', a')\right]^2$

# The Theory: Other tricks

- Batch normalization:
  - Observed in the DDPG paper
- Target networks:
  - Stabilizes training
  - Soft update equation

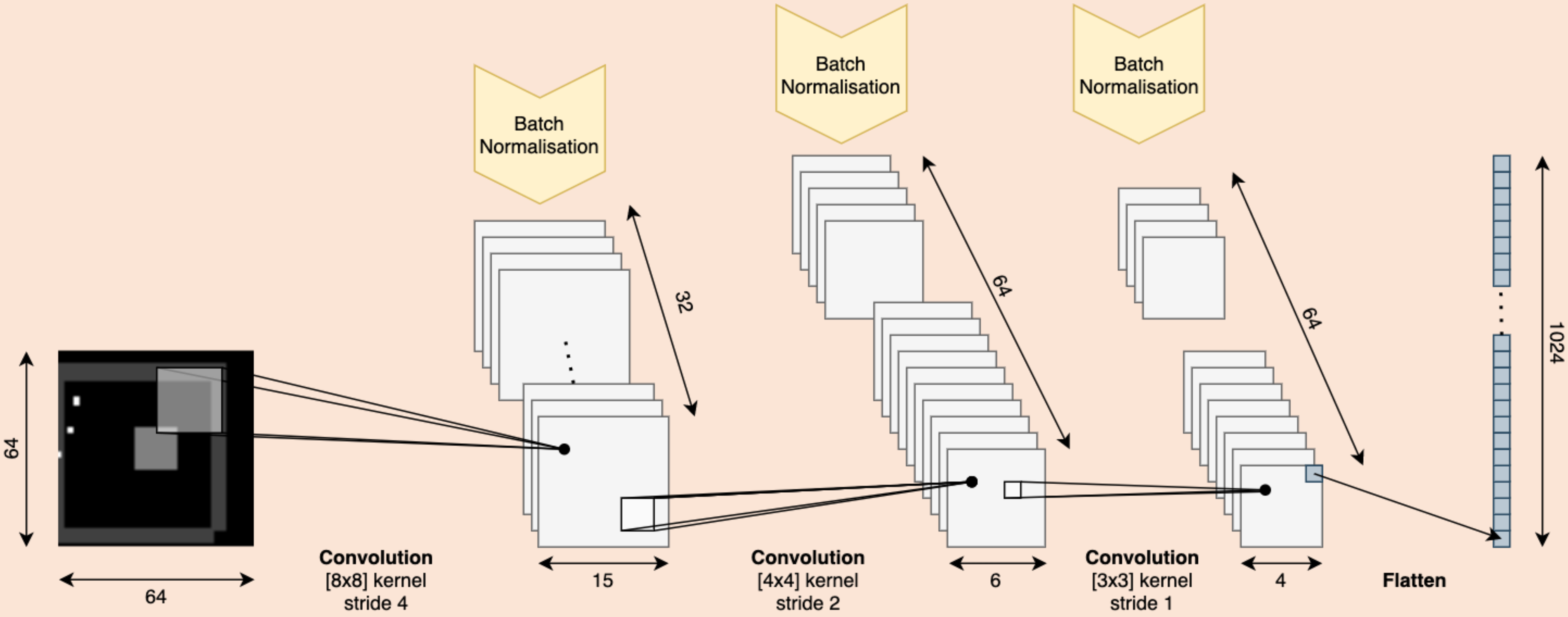
$$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta' \text{ with } \tau \in [0,1]$$

# The Methods: General Network

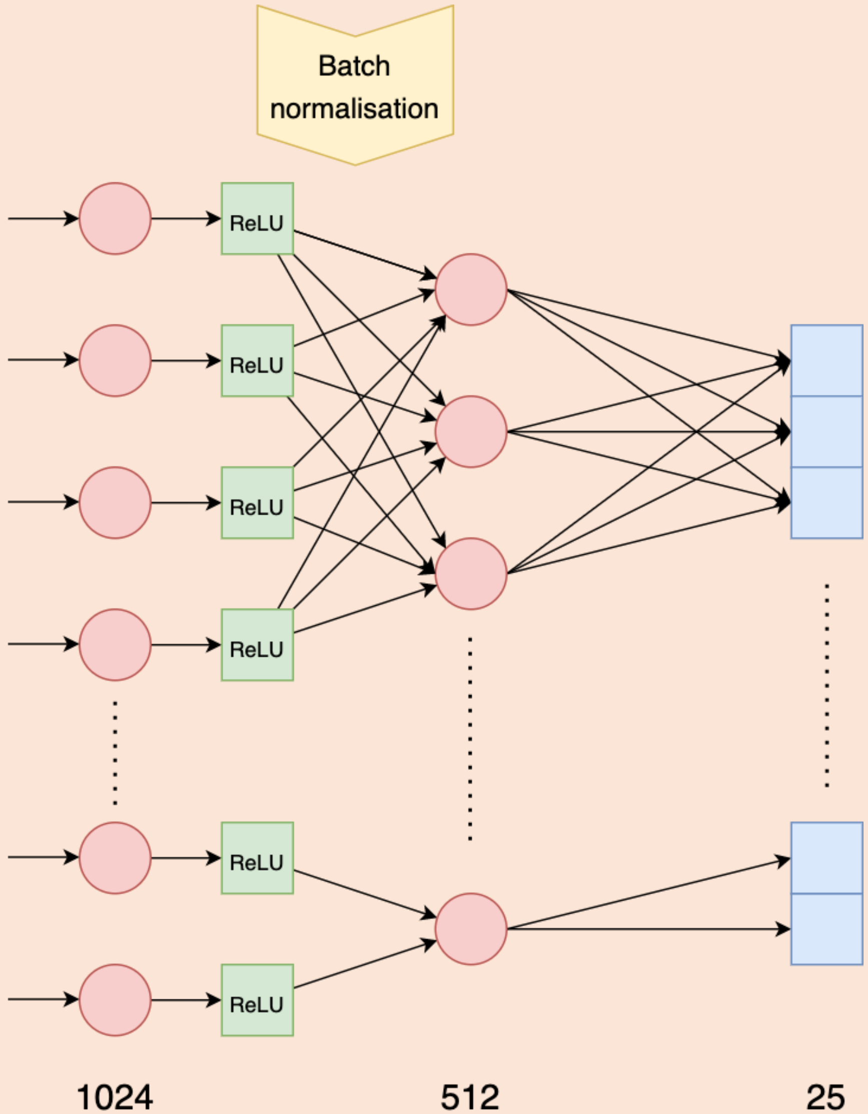


# The Methods: Convolutional Network

- 3 Convolution layers
- Batch normalisation



# The Methods: Multi Layer Perceptron

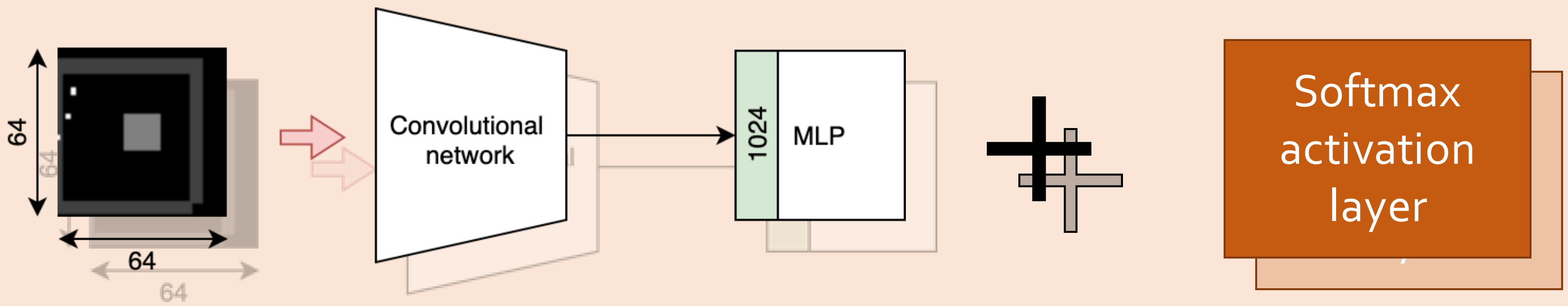



- 2 dense layers
- Batch normalisation

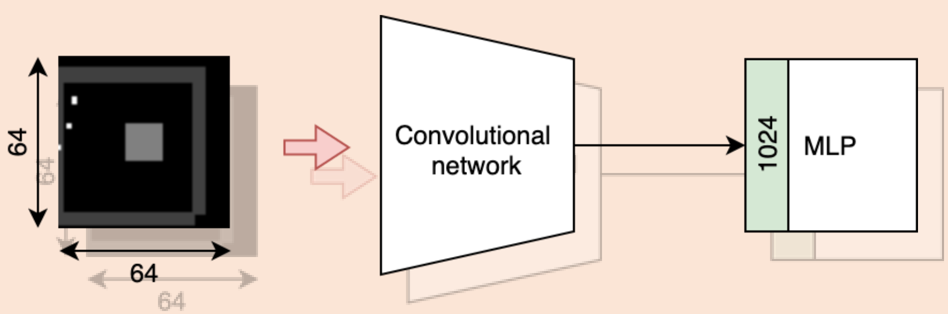
**Different versions for the different algorithms**

# The Methods: A2C

## Actor:



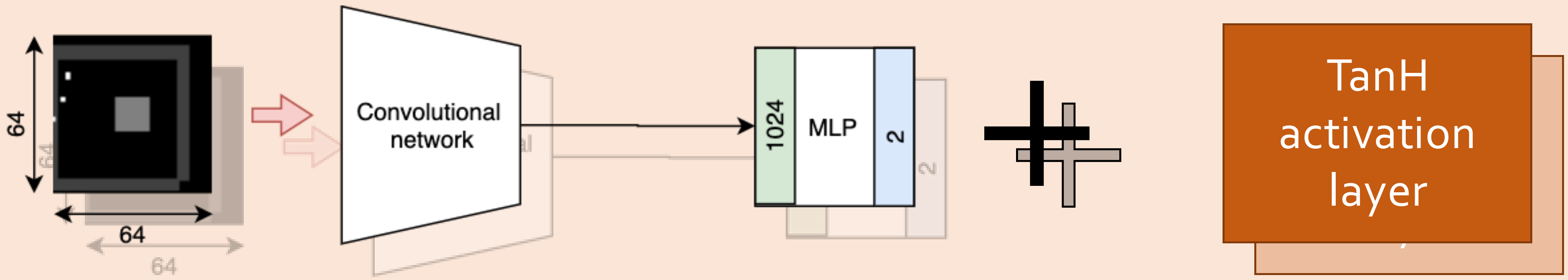

## Critic:



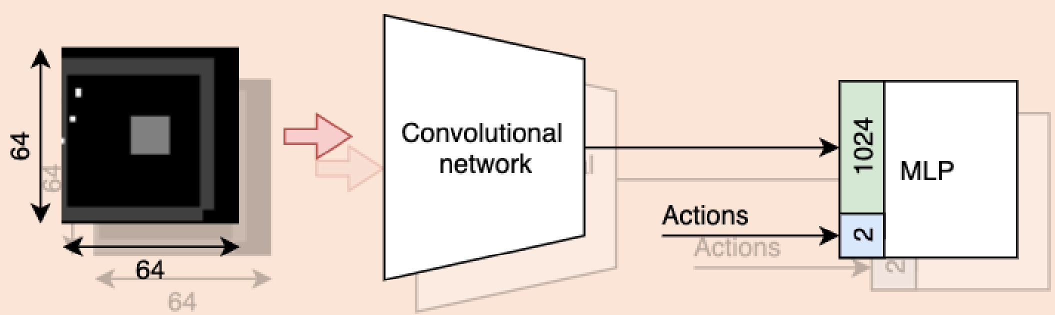

| **Parameters** | **Value** |
|---|---|
| Learning rate | 0.0001 |
| Tau | 0.001 |
| Discount factor | 0.99 |

# The Methods: DDPG

## Actor:

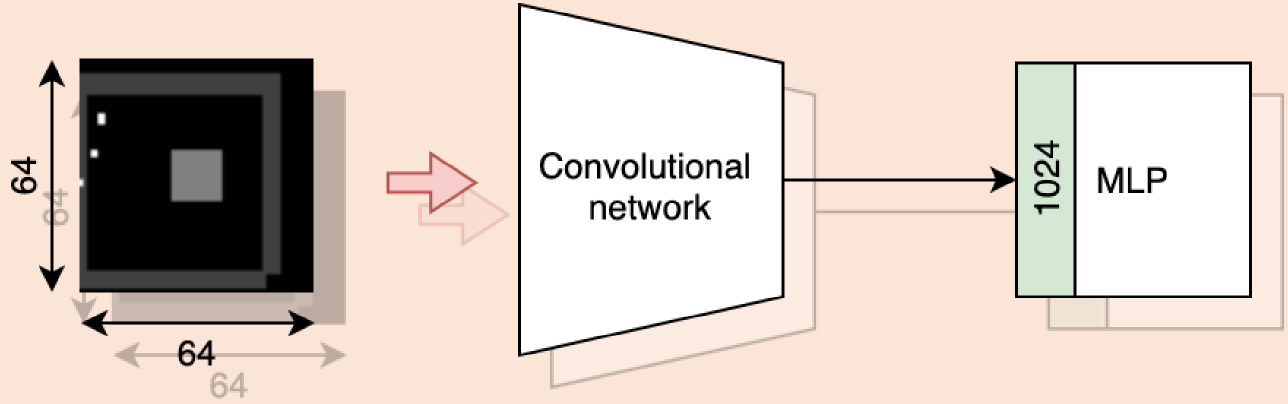



## Critic:




| Parameters | Value |
|---|---|
| Learning rate critic | 0.0005 |
| Learning rate actor | 0.00001 |
| Tau | 0.001 |
| Discount factor | 0.99 |

# The Methods: DDQN




| Parameters | Value |
|---|---|
| Learning rate | 0.0001 |
| Tau | 0.001 |
| Discount factor | 0.99 |

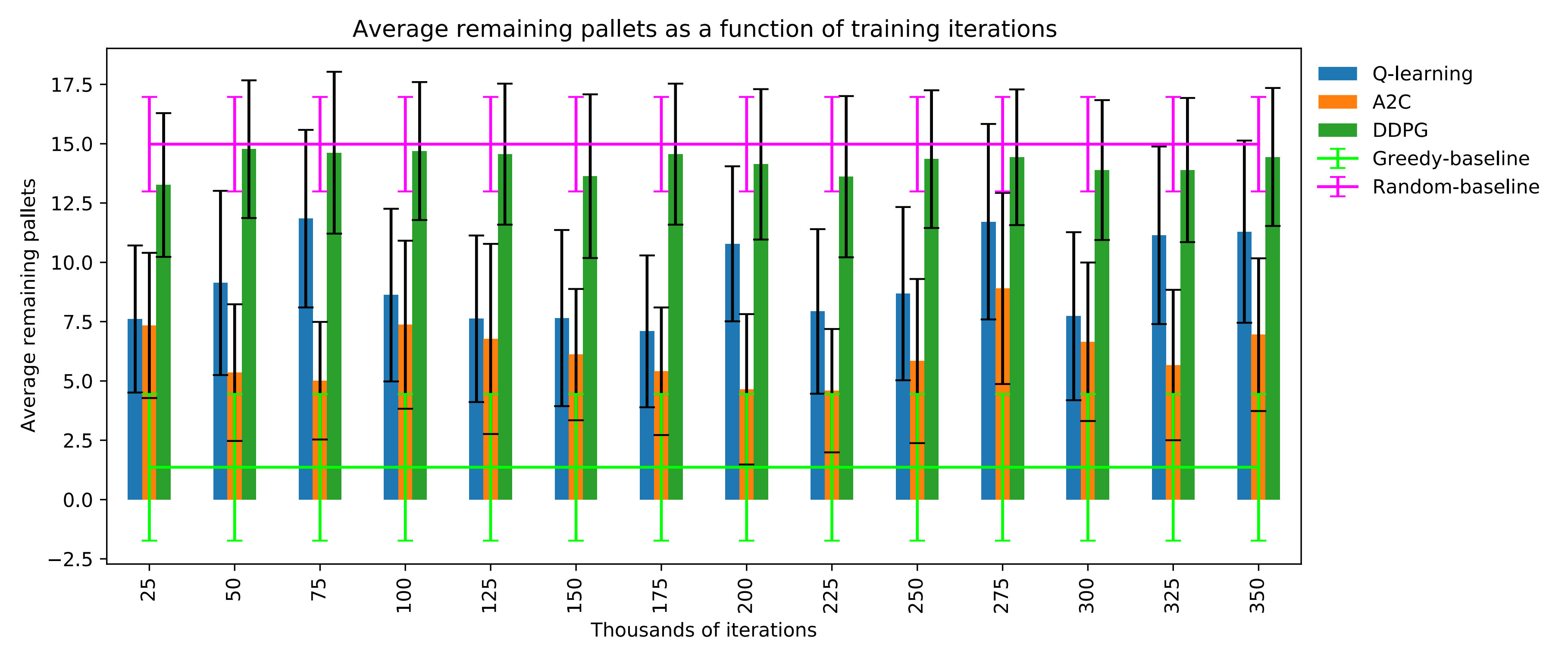
# The Methods: Training

- Random environments
  - Random pellet position
  - Random init position
- Constants:
  - Size of map
  - Number of pellets (20)
- Images of 64 x 64
- Episodes of 2500 steps
- Total of 350 000 frames
- Replay buffer of size 65 000
- Adam optimizer

Hardware:

- *CPU*: I7-6700k 4GHZ (4 cores)
- *GPU*: NVIDIA GTX 1070 (training was ~ 30 % faster)

# The Results

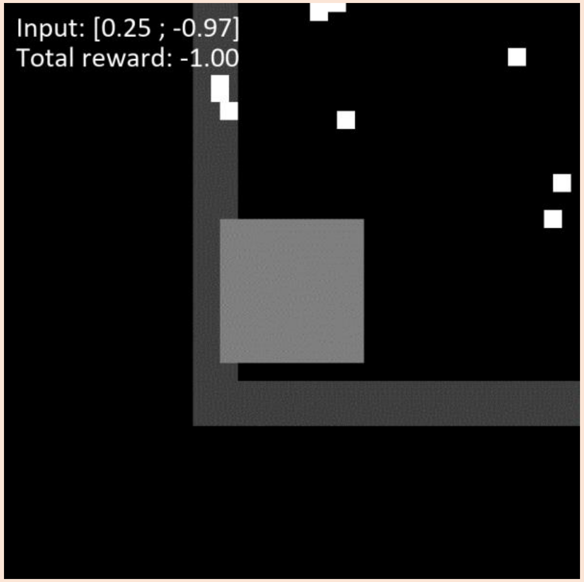


Average remaining pallets as a function of training iterations
Average remaining pallets
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
25
50
75
100
125
150
175
200
225
250
275
300
325
350
Thousands of iterations
Q-learning
A2C
DDPG
Greedy-baseline
Random-baseline

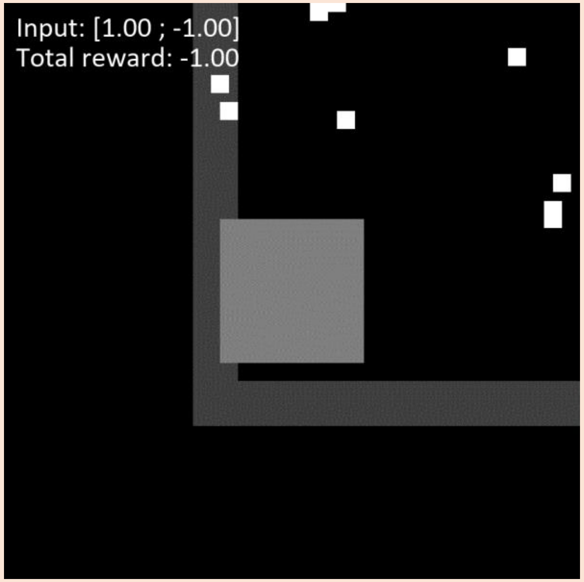
# The Results

Average total reward as a function of training iterations
0
−25
−50
−75
−100
−125
−150
−175
−200
Average total reward
25
50
75
100
125
150
175
200
225
250
275
300
325
350
Thousands of iterations
Q-learning
A2C
DDPG
Greedy-baseline
Random-baseline

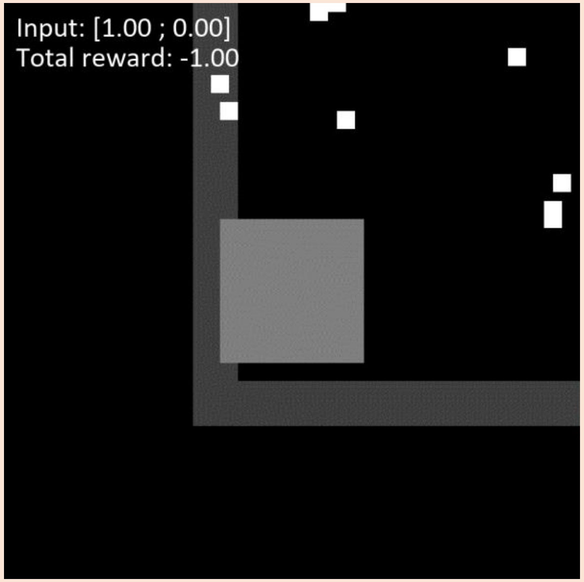
# The Results: example trajectories




Greedy agent




A2C after 3000 steps




A2C after 4000 steps

# The Results: Discussion

- Lack of training
- Simplicity of RL Algorithms
- Complexity of the input
- Size of the replay buffer

# Thank you for your attention