IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UE Introduction to AI

PROJECT

TEAM 2

Final restitution

Minh Triet Vo
Wenjie Fu

SUMMARY

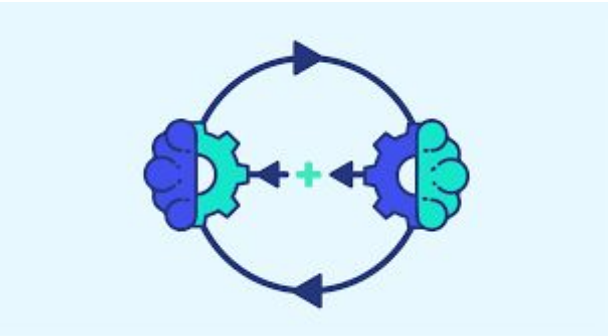
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

RL + Game Theory !

1. Why not supervised Learning ?
2. Why not unsupervised Learning ?
3. RL only -> 53% winning rate?
4. RL + Game Theory

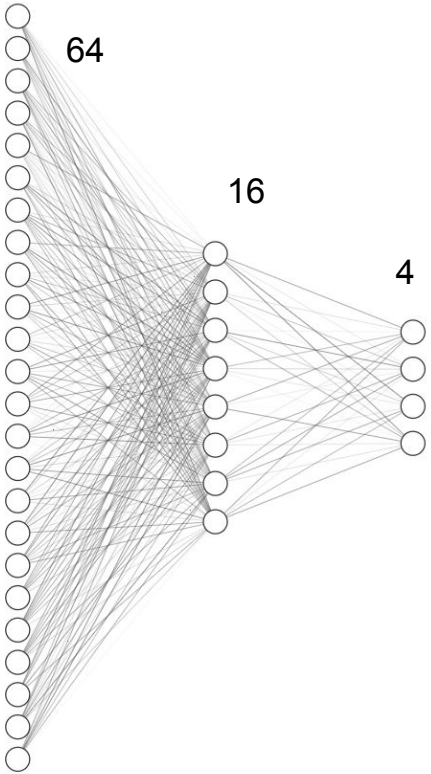
Action at time(t)

$$\begin{cases} \max Q_t(a) & \text{with probability } 1-\epsilon \\ \text{any action (a)} & \text{with probability } \epsilon \end{cases}$$



$$\epsilon = \epsilon_{end} + \left(\epsilon_{start} - \epsilon_{end}\right) * e^{-\frac{\epsilon_{step}}{\text{decay rate}}}$$

Network configuration



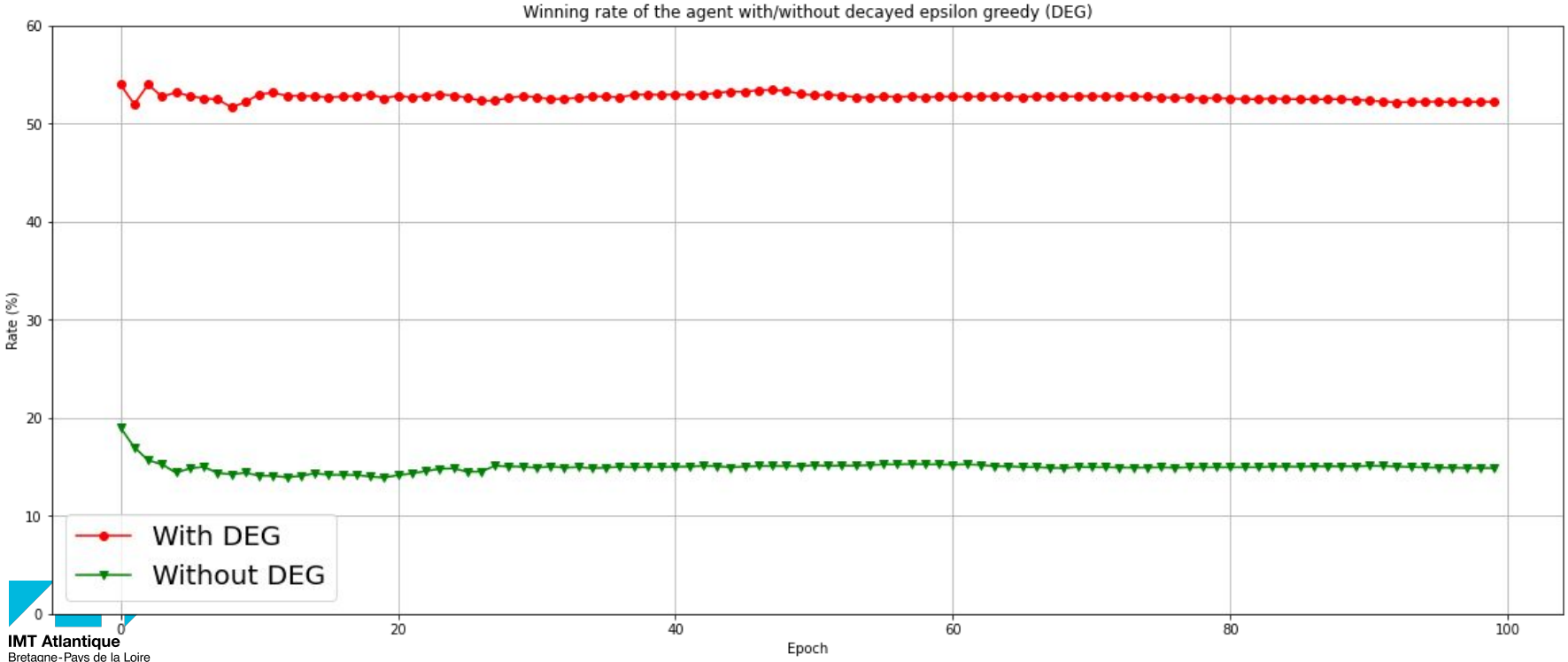
Action choosing **without** epsilon greedy

```
with torch.no_grad():
    Q_t = model(input_curr.unsqueeze(dim=0))[0]
    action_t = torch.argmax(Q_t).item()
```

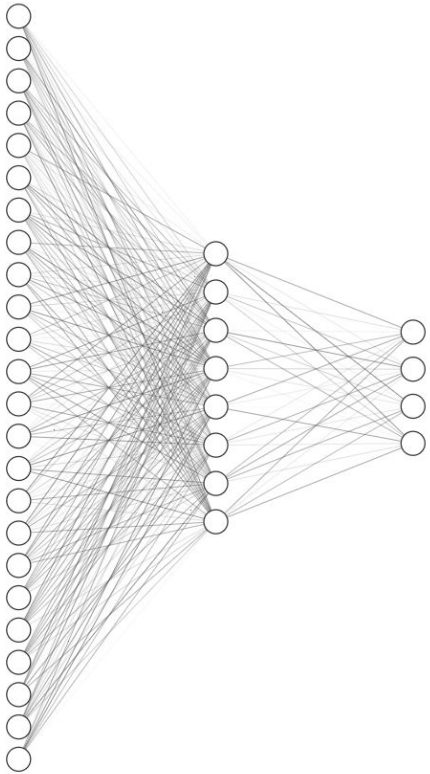
Action choosing **with** epsilon greedy

```
def eps_greedy_action(model, state, eps_start, eps_end, decay_rate, eps_step):
    # This is the function for decayed-epsilon-greedy action-selection implementation
    ## Formulate the decaying of epsilon
    eps = eps_end + (eps_start-eps_end)*np.exp(-eps_step/decay_rate)
    ## Action-selection based on epsilon value
    if np.random.rand() > eps:
        with torch.no_grad():
            Q = model(state.unsqueeze(dim=0))[0]
            ## Pick the next action that maximizes the Q value
            action = torch.argmax(Q).item()
    else:
      ## Pick the next action randomly
        action = np.random.choice(4)
    return action
action_t = eps_greedy_action(model, input_curr, eps_start, eps_end, decay_rate, eps_step)
```

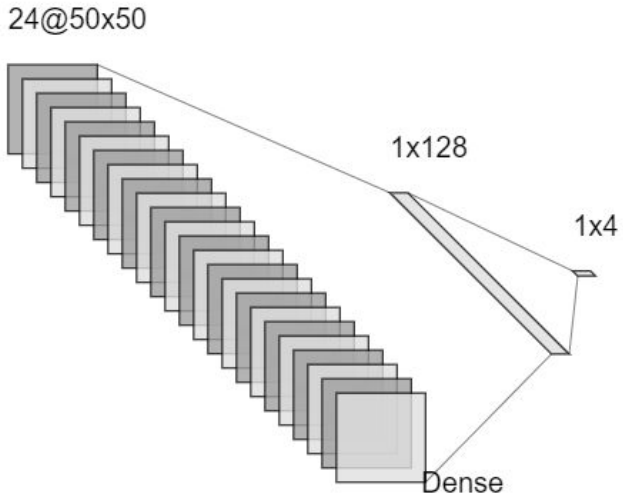
Results


Winning rate of the agent with/without decayed epsilon greedy (DEG)
Rate (%)
Epoch
0
10
20
30
40
50
60
0
20
40
60
80
100
With DEG
Without DEG

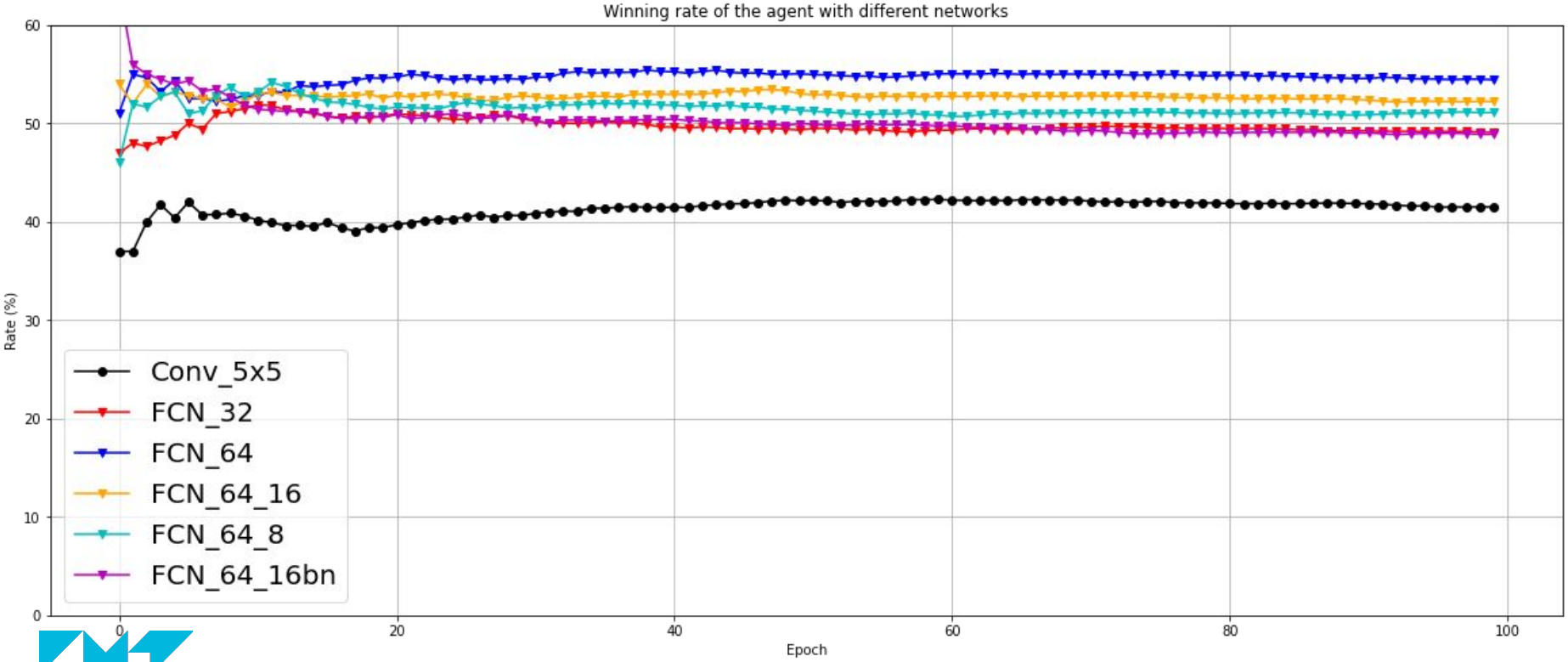
Network configuration



VS



Name	Layers
Conv_5x5	2 conv (size=5, stride=1)
FCN_32	dense (32)
FCN_64	dense (64)
FCN_64_16	dense (64) → dense (16)
FCN_64_8	dense (64) → dense (8)
FCN_64_8bn	dense (64) + batchnorm → dense (8) + batchnorm

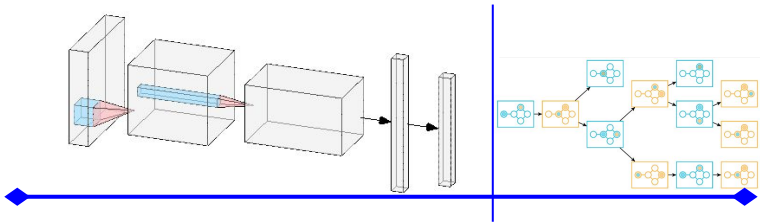

Winning rate of the agent with different networks
Rate (%)
Epoch
0
10
20
30
40
50
60
0
20
40
60
80
100
Conv_5x5
FCN_32
FCN_64
FCN_64_16
FCN_64_8
FCN_64_16bn

How?

→ Put a threshold for the AI to choose its strategy.

Why?

- RL: Good but bounded performance.
- Game theory: unbounded performance but really slow.



?

Name	Winning rate (100 games)			
	7 cheese	10 cheese	12 cheese	13 cheese
Conv_5x5	53%	63%	78%	72%
FCN_32	71%	78%	73%	80%
FCN_64	57%	70%	79%	90%
FCN_64_16	**63%**	**82%**	**91%**	**89%**
FCN_64_8	74%	76%	77%	78%
FCN_64_8bn	64%	69%	78%	81%

Reference:

1. https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/
2. https://aakash94.github.io/Reward-Based-Epsilon-Decay/

Thank you!

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom