SoccerTwos
IFT608-702

Projet d’apprentissage par renforcement

Mohammed Benabbassi
Elio Torquet
Mahutondji Josué Friedman Couthon
Farouk Oussada
Abdou Rahime Daouda

Plan de la présentation

Environnement — Présentation & technologies

MADDPG — Présentation, contribution & résultats

QMIX — Présentation, contribution & résultats

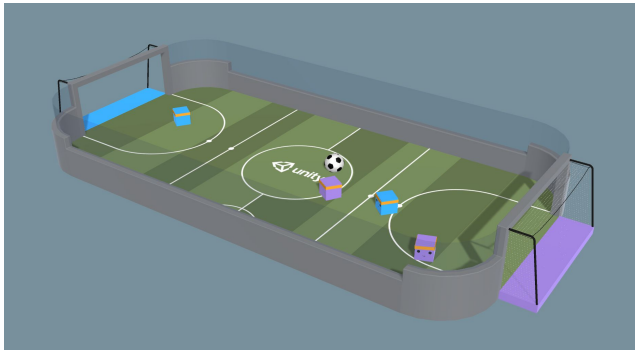
Conclusion — Bilan & perspectives futures

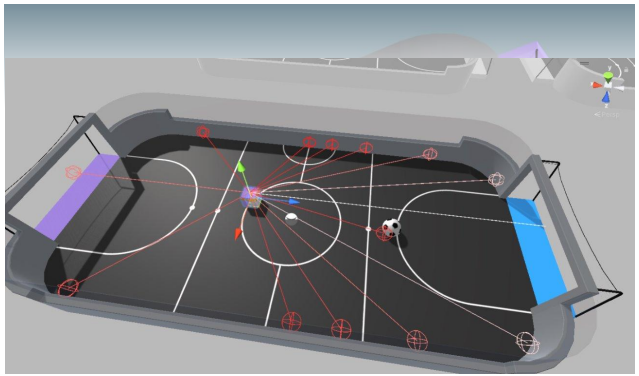
Présentation de l'environnement

- **Situation** : soccer 2 VS 2.
- **Objectifs** : marquer un but rapidement, ne pas encaisser de buts.
- **Capteurs**: plusieurs "lasers" devant et derrière.
- **Actions** : avant/arrière, côtés et rotations.

Forward Motion: UP / DOWN
Sideways Motion: LEFT / RIGHT
Rotation: LEFT/RIGHT

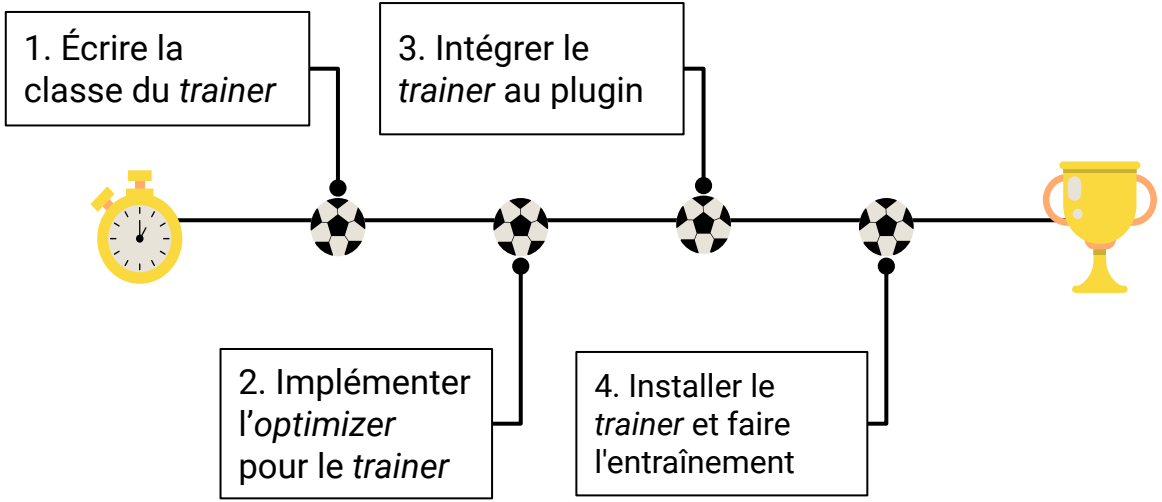
- **Multi-agents** (2 équipes de 2 agents).
- **Compétitif** et **Collaboratif**.





Trainer/Optimizer avec ML-agents

Système de plugin extensible




```
├── mlagents_trainer_plugin
│   ├── __init__.py
│   ├── a2c
│   │   ├── __init__.py
│   │   ├── a2c_3DBall.yaml
│   │   ├── a2c_optimizer.py
│   │   └── a2c_trainer.py
│   └── dqn
│       ├── __init__.py
│       ├── dqn_basic.yaml
│       ├── dqn_optimizer.py
│       └── dqn_trainer.py
└── setup.py
```

Trainer/Optimizer de ML-agents

Avantages

- Intégration **profonde** avec Unity.
- **Haute performance** (optimisé pour Unity).
- Support multimodal (entrées visuelles, audio et autres) → variétés de capteurs.
- **Exemples/tutoriels** fournis pour POCA et autres algorithmes.

Inconvénients

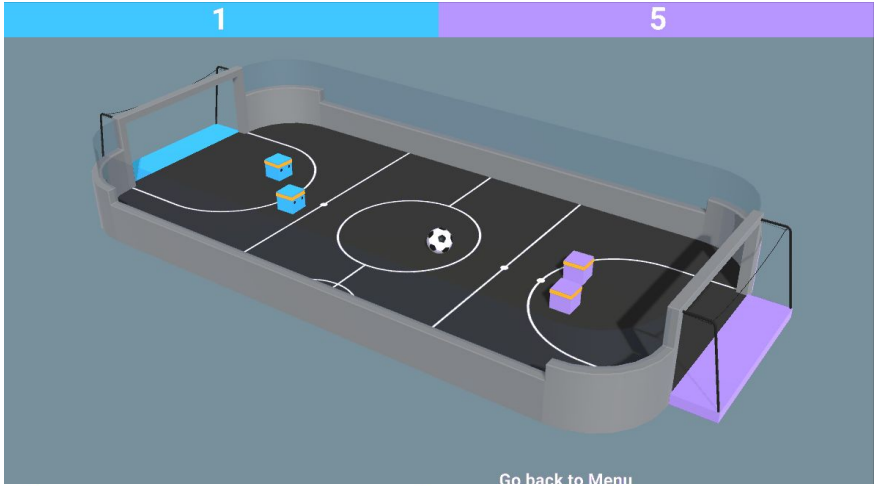
- **Forte dépendance** à Unity.
- Configuration et implémentation **complexes**.
- Débogage **complexe.**
- **Moins flexible.**

Hugging Face

- Cours complet pour débutant en RL*.
- Documentation très détaillée.
- Beaucoup d'implémentations d'algorithmes/modèles déjà faites.
- Possibilité de comparer son modèle (AI vs AI Challenge).

- Impossibilité de modifier les espaces d'observations et d'actions de l'agent
- Impossibilité d'utiliser un *trainer* personnalisé.

*Hands-on - Hugging Face Deep RL Course



AI vs. AI SoccerTwos Leaderboard

In this leaderboard, you can find the ELO score and the rank of your trained model for the SoccerTwos environment.

If you want to know more about a model, just **copy the username and model and paste them into the search bar.**

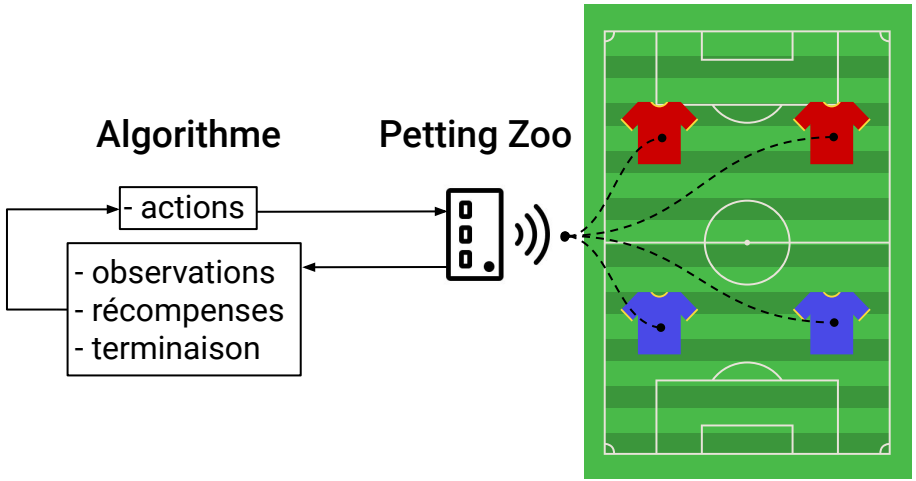
To visualize your agents competing check this demo: https://huggingface.co/spaces/unity/ML-Agents-SoccerTwos

For more information about this AI vs. AI challenge and to participate? Check this

rank	author	model	elo	games_played
1	Agog	Impatient	1803.958502919042 8	252
2	Agog	Soccer	1756.330411881660 2	874
3	atorre	poca-SoccerTwos-70M	1740.505443083147 8	1025
4	ZhihongDeng	poca-SoccerTwos	1731.656739671013 2	802
5	jinhu2659	poca-SoccerTwos	1712.975395772759	502

PettingZoo

- Interface de contrôle standardisée pour du RL Multi-Agents.
- Fonctionnement de l'interface (méthode *step(action)*).
- Intégration avec Unity via un Wrapper.
- Implémentation plus conventionnelle des algos.
- Beaucoup plus documenté et utilisé.



Modifications apportées à l'environnement

- Observation "**totale**"
- **Relatives** à chaque agent
- **16** entrées (au lieu de 336)

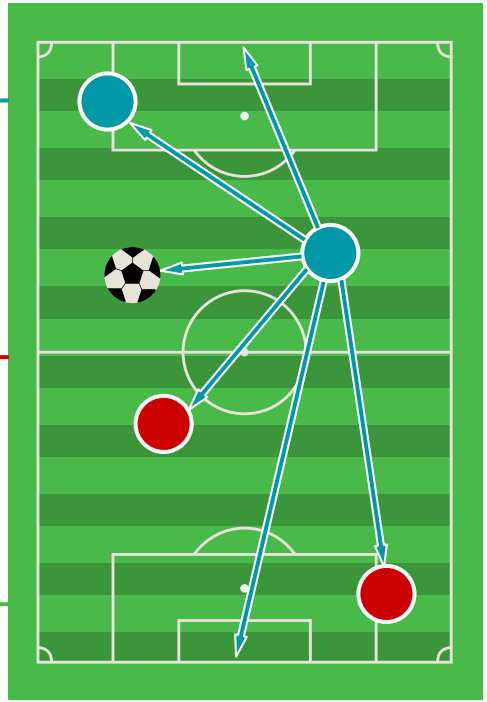
Observations

- **MA-DDPG :** 3 actions continues
- **Q-Mix :** 3 espaces d'actions discrets

Actions

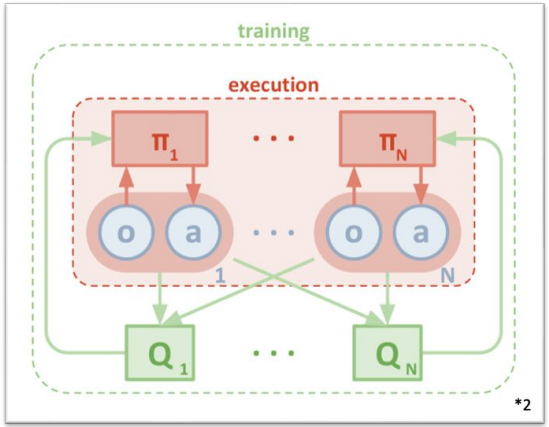
- **Buts** rapides
- **Touches de balle** dans la bonne direction (décroît dans le temps)

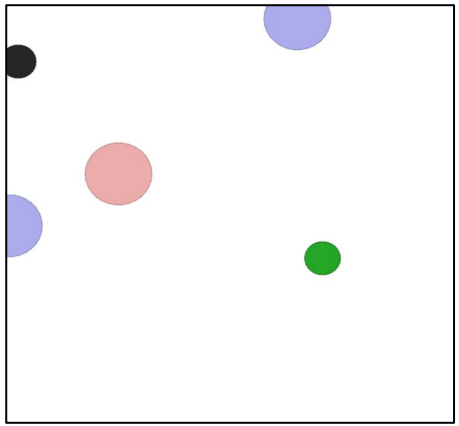
Récompenses



MADDPG - Point de départ

- **Type :** Policy/value gradient.
- Entraînement **centralisé**, exécution **décentralisée.**
- Adapté pour la **collaboration** et la **compétition.**
  - ➢ Un critique "**omniscient**" distinct pour chaque agent.
  - ➢ Récompense distribué par agent (objectifs distincts).

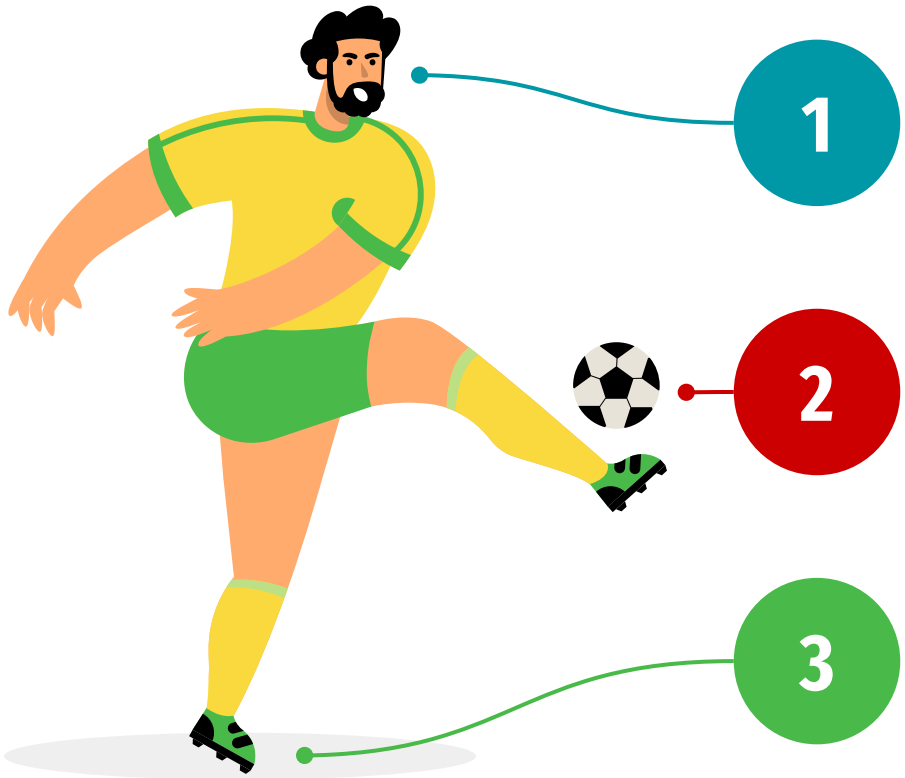




Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments (Lowe & al. 2017)

MADDPG-PyTorch par Shariq Iqbal

MADDPG - Contribution et améliorations



Exploitation

- **Replay buffer**
- **Exploration** par OUNoise
- **Gumbel-Softmax** pour les actions discrètes

Adaptation

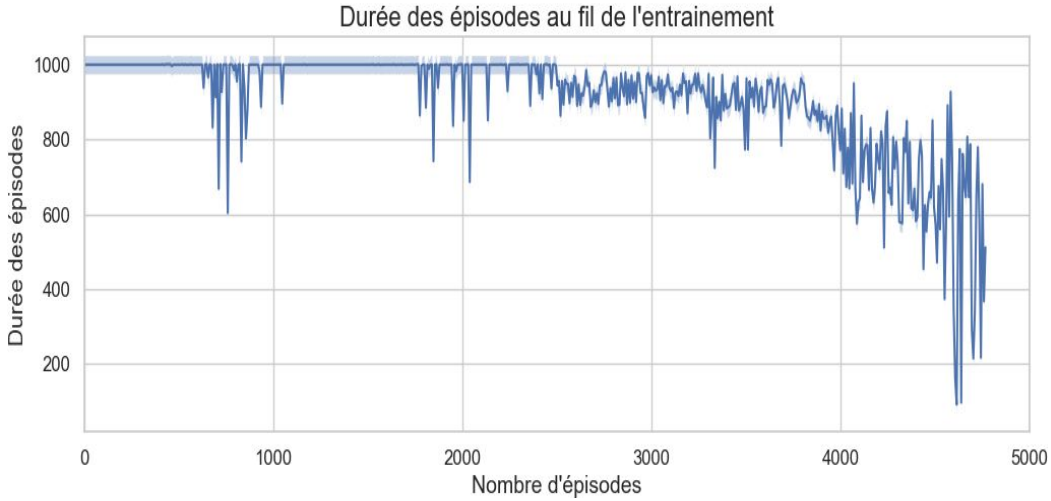
- Exécution d'**instances parallèles** de l'environnement
- **Sauvegarde/Chargement** des modèles
- **Parallélisation des calculs** via CUDA

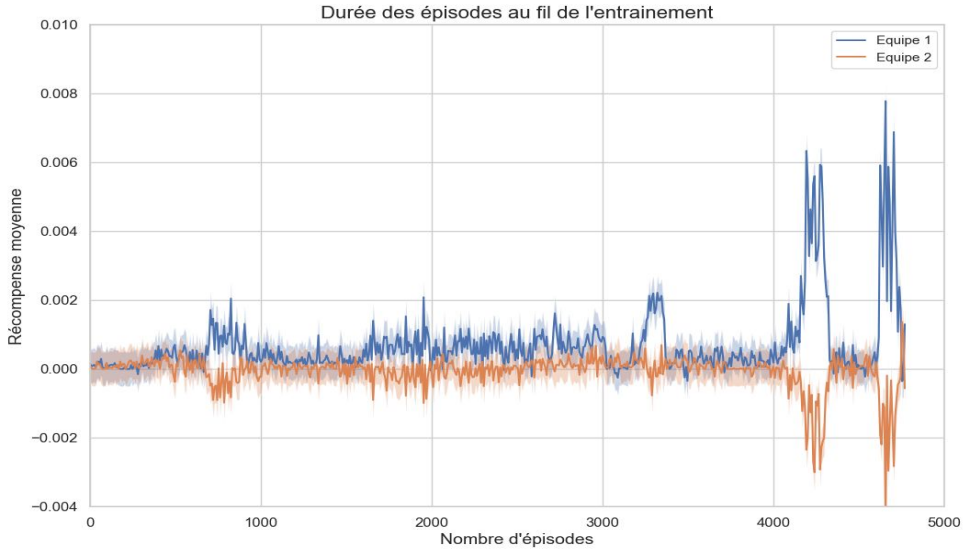
Création

- **Communication avec Unity** via PettingZoo
- **Scaling des paramètres** de l'environnement pendant l'entraînement
- **Paramétrisation** via yaml

MADDPG - Résultats

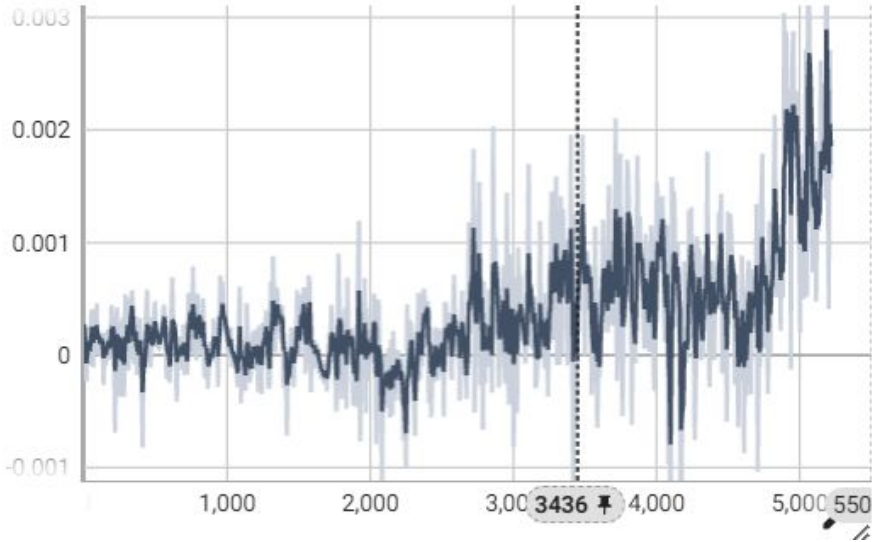
Paramètres d'entraînement	
Environnement :	
Nombre d'instances	8
Nombres d'épisodes	5000
Time scale	20
Ball touch : initial → final	2.0x → 0.0x
Modèle :	
Step par update	256
Hidden units	128
Learning rate actor / critic	3e-4 / 1e-3
Tau	0.001
Gamma	0.9
Batch size	1024
Bruit : initial → final	5.0 → 0.0



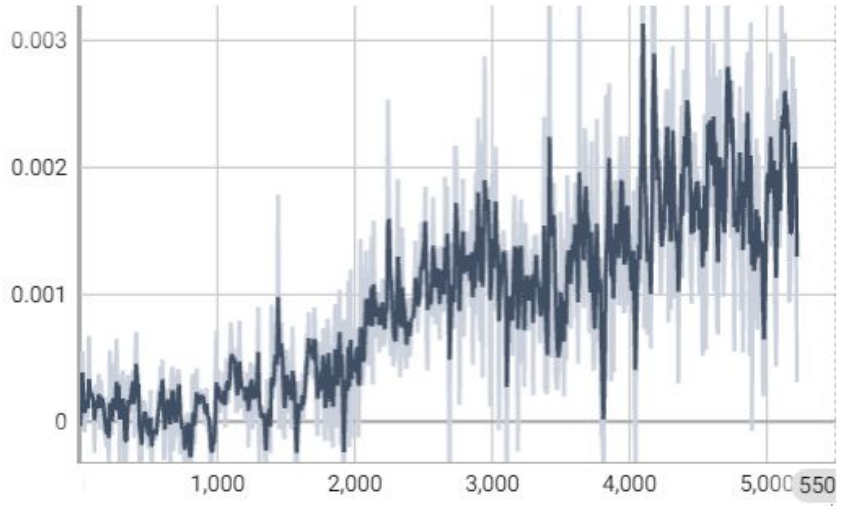


MADDPG - Résultats

Team 1



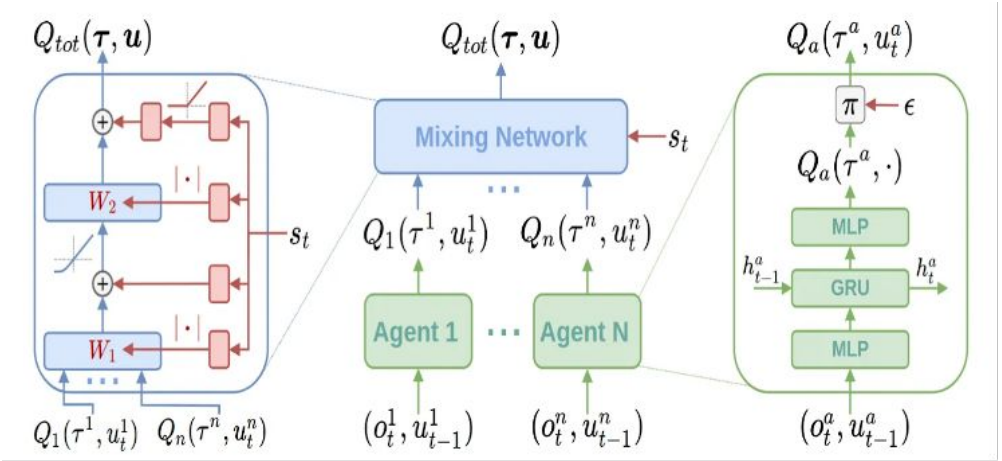
Team 2



Récompense moyenne des équipes sur ~5000 étapes.

QMIX- Point de départ

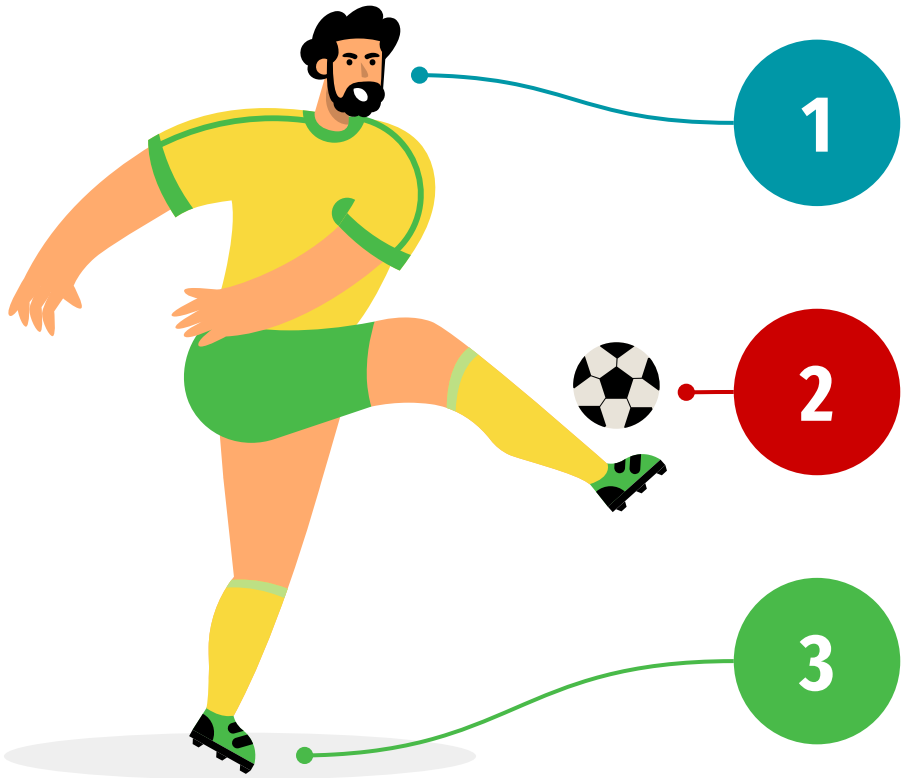
- **Type:** État/action-valeur.
- Entraînement **centralisé**, exécution **décentralisée.**
- Adapté pour la **compétition** & la **collaboration.**
  - ➢ Récompense distribué par agent.
  - ➢ Réseau de mixture.
- **Réseau d'un agent**
  - ➢ Prend l'observation et action précédente et calcule l'utilité (Qa).
- **Réseau de mixture**
  - ➢ Prend les Qa de tous les agents et calcule la Qtot.



Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning (Rashid, T & al. 2018)

QMIX PyTorch par Yang Chen

QMIX- Contribution et améliorations



Exploitation

- **Sauvegarde/Chargement** des modèles.
- **Parallélisation des calculs** via CUDA.
- Communication agents/réseau de mixture.

Adaptation

- Remplacement des **MLP** par **RNN**.
- **Initialisation des poids** au niveau des RNs.
- Architecture du **réseau de mixture** (inputs, layers, outputs).

Création

- **Communication avec Unity** via PettingZoo.
- **Paramétrisation** & **généralisation** via yaml.
- Gestion des actions **multi-discrètes**.
- **ReplayBuffer** adapté pour RNN.
- Beaucoup de **refactoring** & **clean up.**

QMIX- Progrès

- Adaptation des **actions multi-discrètes.**
  - Option 1: formuler toutes les combinaisons d'actions possibles.
  - Option 2: passer 3 fois le RN (pour chaque type d'action).
- Finaliser l'adaptation du **Replay Buffer**.
  - Suite à l'utilisation du **RNN**.
  - Régler des problèmes stockage (CUDA vs CPU).

Mouvement longitudinal	Rien faire	avant	arrière
Mouvement latéral	Rien faire	gauche	droite
Rotation	Rien faire	Anti horaire	horaire
	0	1	2

3x3 →

0	0	0
(...)		
2	2	2

Bilan

Points Positifs

- **Algorithmes intéressants et pertinents** vis à vis du problème.
  - Multi-agents, collaboratif, compétitif
- **Implémentation complète et portable**.
  - Adaptation à d'autres environnements Unity ou PettingZoo.
  - Documentation de notre code.
- **Manipulation concrètes** de l'environnement
  - Meilleur compréhension des environnements ml-agents
- **Parallélisation des calculs** sur GPU.
  - Découverte et utilisation Torch

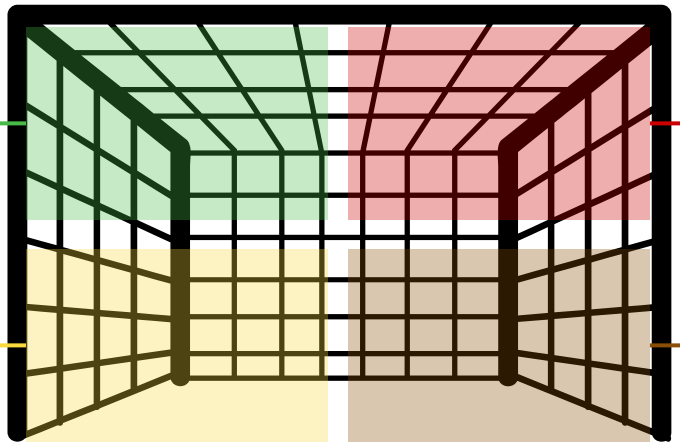
Points Négatifs

- **Complexité de Unity ML-agents**
  - Très peu documenté (des *Markdown* bien cachés).
  - Tutoriels de ML-agents ne sont pas à jour.
  - Difficile à installer et démarrer le projet.
- Algorithmes **récents** et **complexes**.
  - Requièrent plus de temps/ressources pour les entraîner.
  - Moins documenté, moins utilisé.
- **Mieux** évaluer la contrainte du **temps** vs l'**ampleur** du projet.
- Commencer par la base avec un environnement plus **simple**.
- **Disponibilité** des ressources matérielles (**GPU**).

Perspectives futures

Faire plus d'expérimentations sur l'entraînement (recherche hyperparams)

Obtenir des résultats pour Qmix



Tester la portabilité des implémentations sur d'autres environnements

Tester sur soccer twos Vanilla (comparer avec les autres implémentations).

Merci