

# Projet 7 Participez à une compétition Kaggle !

## présentation de la compétition

Kore 2022<sup>[1]</sup> est une compétition Kaggle sans base de données, où l'objectif est de développer un algorithme qui gagne au jeu Kore.

Kore est comme un jeu de stratégie de compétition à deux joueurs, des ressources (kore) sont dispersées sur le plateau, et l'objectif est d'envoyer des flottes de vaisseaux pour en récupérer un maximum et les rapporter à la base. Le but du jeu est de finir la partie avec le plus de ressources ou de battre l'adversaire avant le temps imparti.

Les règles sont décrites en détail sur la page de la compétition<sup>[2]</sup>.

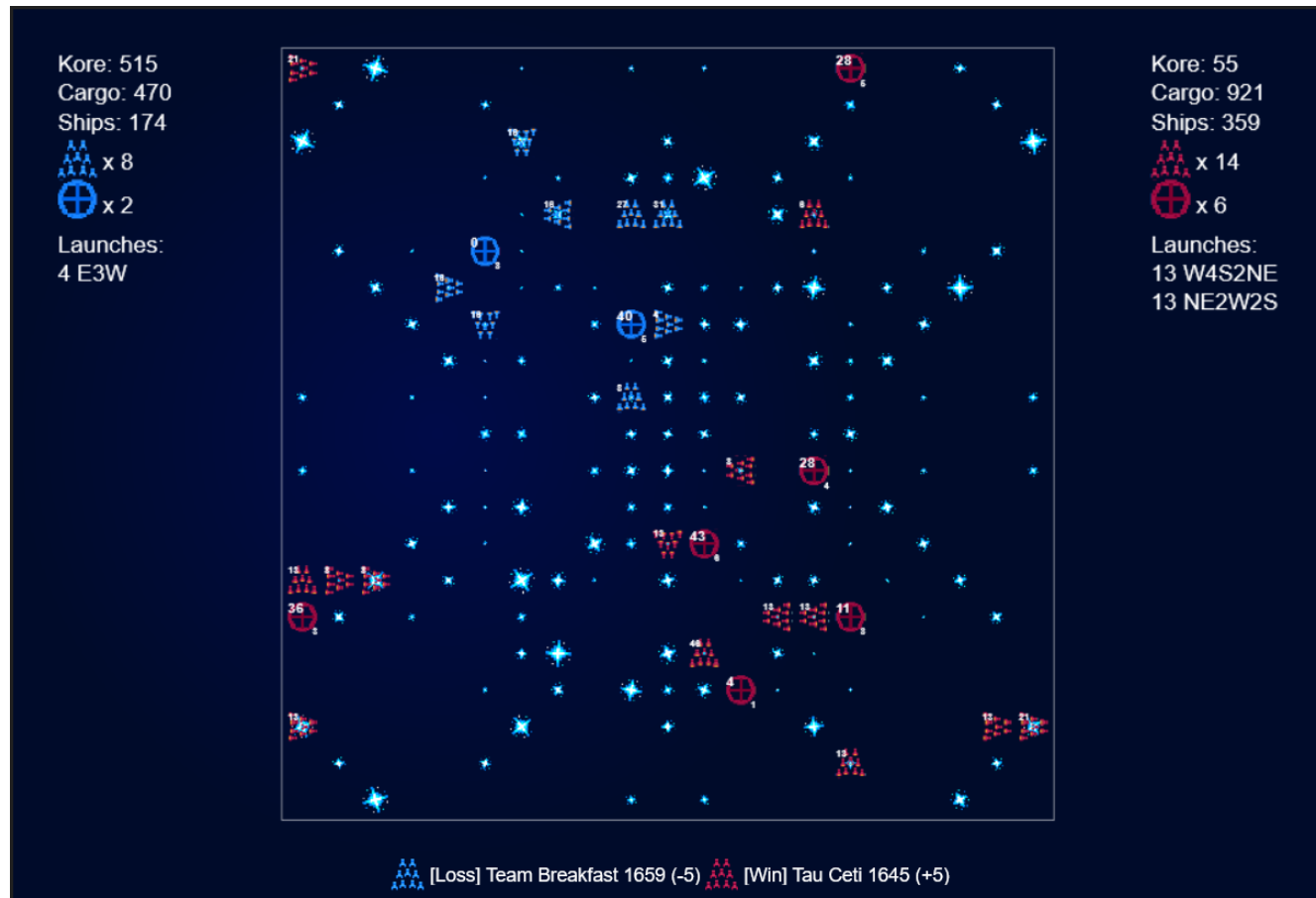
En bref, il faut savoir que la partie est découpée en 400 tours de jeu et pour chaque tour, les deux joueurs doivent choisir l'action que fera chacune de leur base spatiale respective.

chaque base spatiale peut choisir entre :

- { construire des vaisseaux (ce qui coûte des kores)
- { envoyer une flotte de vaisseaux
- { ne rien faire

Dans le cas du choix "envoyer une flotte de vaisseaux" il faut :

1. { choisir un nombre de vaisseaux à envoyer
2. { définir un plan de vol résumé en une suite de caractères qui indique la direction (Nord, Sud, Est, Ouest) la longueur de cette direction, et éventuellement la création d'une nouvelle base.



Voici une partie en cours. Le joueur bleu à plus de kore, mais à en voir le plateau et le nombre de vaisseaux, il se fait submerger par les rouges.

## première recherche et exploration

Dans un premier temps il m'a fallu étudier comment appliquer les technologies de machine learning dans le cadre de cette compétition, car appliquer un algorithme en dur ne rentrait pas dans le cadre de la formation. Il aurait tout de même eu ses propres limitations, qui ne l'aurait pas rendu meilleur qu'un vrai modèle ayant appris le jeu.

La première base fondamentale que j'ai prise vient du notebook [Reinforcement Learning baseline in Python](#)<sup>[3]</sup> qui donne toutes les clés en main pour faire une première baseline de modèle par Reinforcement Learning (RL).

Donc c'est vers le RL que je me suis dirigé pour participer à cette compétition.

Cette technique consiste à partir d'un modèle naïf qui joue aléatoirement. Et au fur et à mesure des parties qu'il joue contre un adversaire (idéalement lui-même), il apprend les meilleures stratégies et améliore les résultats jusqu'à battre les plus forts.

Le modèle le plus populaire qui utilise cette technique c'est AlphaGo qui a battu les meilleurs joueurs de go.

Pour prendre en main l'API de Kore je me suis beaucoup référé à la série de notebook dont le premier est [Kore Intro I: The Basics](#)<sup>[4]</sup>.

Ils permettent de bien comprendre comment interagir avec l'interface proposée par Kore.

L'un des participants a mis en ligne son application proposant toute une interface pour suivre une partie. Son application s'appelle [koreye](#)<sup>[5]</sup>.

Son interface m'a bien aidé pour faire le suivi des parties à fin de trouver les possibles manques à gagner, et autres mauvais coups.

Pour finir il me fallait une référence pour déterminer si mon modèle progressait réellement ou non face à la concurrence, pour ça j'ai choisi de confronter mon modèle à l'agent proposé dans le notebook [Kore Beta 1st place solution](#)<sup>[6]</sup> qui propose donc un agent codé en dur qui obtient de très bonnes performances.

Ce qui en fait même un agent trop dur à battre et du coup un excellent objectif, battre cette adversaire !

## Shuntaro Tanaka

Kore: 18      Cargo: 303  
Shipyards: 1      Ships: 100  
Fleets: 7      Timebank: 60.000s

Shipyards

Fleets

Logs

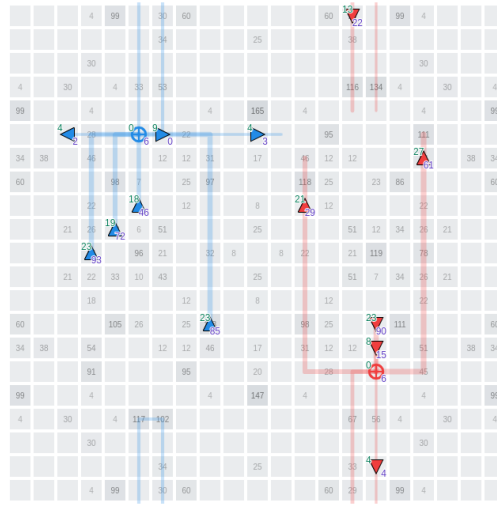
0-1

Cell: (5, 15)

Ships: 0

Spawn: none (max 6)

Launch: 9 ENBWS



4x 75 / 366

## aDg4b

WINNER BY ELIMINATION

Kore: 2      Cargo: 223  
Shipyards: 1      Ships: 96  
Fleets: 6      Timebank: 60.000s

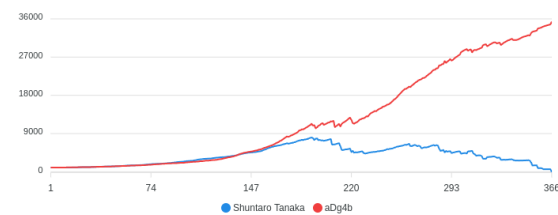
Shipyards

Fleets

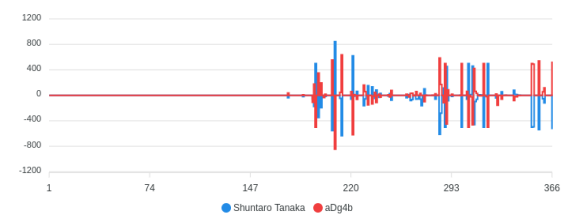
Logs

57-1	Plan: none
Cell: (15, 7)	Direction: south
Ships: 23	Kore: 90
64-2	Plan: 1S9W
Cell: (17, 14)	Direction: north
Ships: 27	Kore: 61
65-1	Plan: 2S8E
Cell: (12, 12)	Direction: north
Ships: 21	Kore: 29
68-2	Plan: 4N9E
Cell: (14, 20)	Direction: south
Ships: 13	Kore: 22
70-2	Plan: none
Cell: (15, 6)	Direction: south
Ships: 8	Kore: 15
71-1	Plan: 6N
Cell: (15, 1)	Direction: south
Ships: 4	Kore: 4

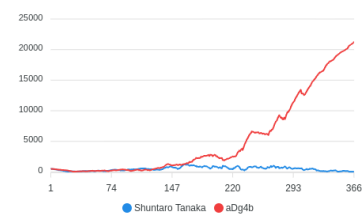
## Total asset worth



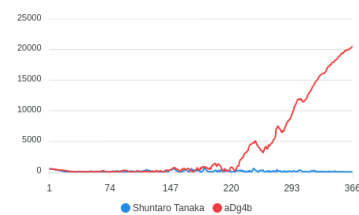
## Kore gains/losses from combat (1 shipyard = 500 kore)



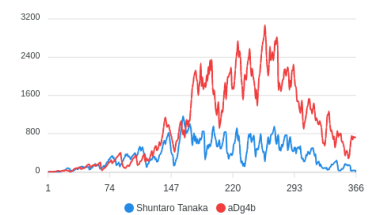
## Total kore



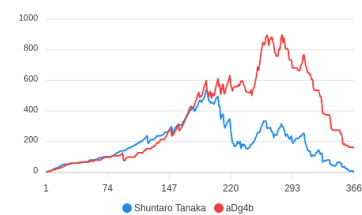
## Stored kore



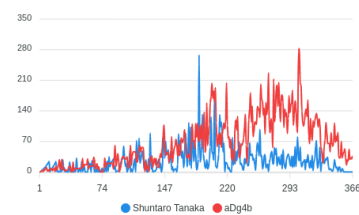
## Kore in fleets



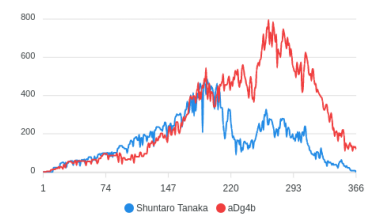
## Total ships



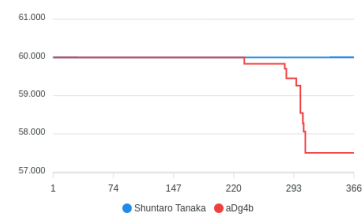
## Ships in shipyards



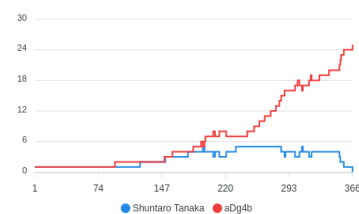
## Ships in fleets



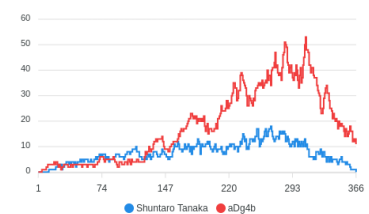
## Remaining overage time



## Total shipyards



## Total fleets



Voici l'interface de Koreye

## Développement du modèle

Le modèle que j'entraîne et que je soumetts est un modèle par Reinforcement Learning (RL), de ce fait, il y a principalement 3 éléments cruciaux à faire évoluer à fin de perfectionner le modèle.

1. Sous quelle forme le réseau de neurones reçoit les informations sur le plateau de jeu.
2. Comment exploiter les informations que le réseau de neurones donne en retour
3. Comment récompenser à la juste valeur (ni trop ni pas assez) le réseau de neurones pour l'action qu'il a effectuée.

Essentiellement toute la difficulté du RL réside dans ces 3 étapes à ajuster comme il le faut.

Il faut savoir que les variables qui rentrent ou sortent du réseau de neurones sont des valeurs continues de -1 à 1, donc tout ce qui rentre doit être normalisé pour rentrer dans l'intervalle, et tout ce qui en sort doit être exploité ainsi.

Pour le circuit de récompense, il n'y a pas de bornes, à savoir qu'une valeur négative sera interprétée comme une sanction (pour un mauvais coup) et une valeur positive sera interprétée comme une récompense (pour un bon coup)

## Etat de la baseline

La baseline tel qu'elle était proposée avait déjà ces 3 étapes configurées arbitrairement, je vais décrire brièvement comment elle était pour présenter de quoi je suis partie, afin de mieux expliquer l'évolution que j'y ai ajoutée.

### Du plateau au réseau

Pour décrire l'état du terrain, la baseline fournissait ses différentes variables :

1. pour chaque cellule du plateau la quantité de kore
2. pour chaque cellule du plateau la quantité de flotte adverse ou alliée
3. la direction de la flotte
4. pour chaque cellule du plateau s'il y a une base adverse ou alliée
5. la quantité de kore que le joueur possède
6. la quantité de kore que l'adversaire possède
7. l'avancement de la partie (le nombre de tours joué)

### Du réseau aux actions

La baseline était très basique sur cette étape, car il n'y avait le choix qu'entre :

1. construire des vaisseaux
2. envoyer une flotte de vaisseaux dans un axe cardinal.

ce qui limite grandement les possibilités.

Dans le cas de plusieurs bases spatiales l'action donnée sera faite identiquement pour chacune, donc les actions ne sont pas propres à la base.

## Circuit de récompense

La baseline était imaginée de sorte à ce que tout éléments du jeu soient convertis en équivalent kore, par exemple un vaisseau coûte 10 kores à la construction, donc sa valeur ajoutée dans le circuit de la récompense vaut 10.

et c'est pareil pour le reste.

- la valeur des bases spatiales
- la valeur des kore dans le cargo (encore qu'elle vaut moins que les kore à la base car elle ne sont pas assurés)

par contre la valeur des vaisseau et base décroissait progressivement au cours de la partie, alors que la valeur du kore croît inversement.

pour mieux récompenser le fait d'avoir des vaisseaux au début, et du kore à la fin.

## Evolutions apporté

j'ai malgré tout apporté de nombreuses modifications à la baseline, je vais ici en décrire les principaux.

1er chose à savoir le modèle que fait tourner la baseline demande les instructions au réseau de neurones chaque tour, et l'action donnée est la même pour tous les bases spatiales ce qui limite les possibilités.

ce que j'ai fait c'est de demander les instructions au réseau de neurones par base spatiale par tour, comme ça, chaque base à une instruction personnalisée.

### Du plateau au réseau

pas beaucoup de modifications ont été effectués sur cette partie, j'ai principalement adapté pour que ce soit propre à la base spatiale donc

- le nombre de vaisseaux dans la base spatiale
- le nombre de vaisseaux en tout

### Du réseau aux actions

en plus des précédentes actions possibles j'en ai ajouté deux autres à savoir.

- la possibilité de faire plan de route qui trace un rectangle à partir de la base
- la possibilité de créer une base autre part sur le terrain

## circuit de récompense

le circuit de la récompense a été grandement modifié pour mieux récompenser les actions qui ne procure pas de récompense directe, mais qui tend à la victoire.

par exemple, pour la baseline après seulement un quart de la partie, le modèle et mieux récompensé à garder les ressources que de créer de nouveaux vaisseaux, alors qu'il faut nécessairement plus de vaisseaux pour collecter des ressources.

moi j'ai préféré sanctionner le modèle quand il a moins de X vaisseaux en service, X que j'ajuste pour que ce soit suffisamment pour collecter les ressources, et pas trop non plus.

## Résultats et retour d'expérience

une chose principale à retenir, c'est que le Reinforcement Learning c'est dur. avec toutes les améliorations apportées, et de nombreuses heures d'apprentissage dans de multiples configurations, mon score final obtenu atteint à peine les 300, et je tourne autour de la place 370/450. il est difficile de savoir quel est le réel souci, qu'est-ce qui fait que ça fonctionne mal.






une chose est sur, j'ai encore des difficultés avec la phase d'apprentissage, car au cours de l'apprentissage le score cumulé de récompense n'augmente pas beaucoup, tout comme la fonction loss ne décroît pas, voir augmente parfois.

## Ressources en ligne

Pour ce projet, j'ai publié un repository de mon code sur GitHub par [ce lien](#) voici [mon compte kaggle](#). On peut me retrouver sur [le leaderbord](#) de Kore 2022

## Bibliographie

---

1. [Kore 2022](#) 
2. [Kore Rules](#) 
3. [Reinforcement Learning baseline in Python](#) 
4. [Kore Intro I: The Basics](#) 
5. [Koreye](#) 
6. [Kore Beta 1st place solution](#) 