Reinforcement Learning for truck Eco-driving: serious Game and Driving assistance system

Pierre-Yves BOISBUNON

Philippe CARRÉ

Anne Sophie CAPELLE-LAIZÉ

Mohamed FASSIH

Juin 2022

Sommaire :

Introduction :

STRADA :

- Spécialisée dans l'édition de logiciels de transport routier.
- 5 000 clients, 8 000 véhicules connectés et 100 000 conducteurs.
- Solutions :
  - ***TIME*** *: Logiciel central de gestion, Time recueille et archive les données des cartes conducteurs et celles des différents tachygraphes.*
  - ***Tracking*** *: Logiciel central de pilotage, il permet le traitement de l'information et le suivi de flotte en temps réel.*
  - ***TMS*** *: Logiciel central d'organisation, il permet de faire l'accompagnement dans le pilotage de votre activité transport.*

STRADA dispose d'accès à une énorme quantité de données.

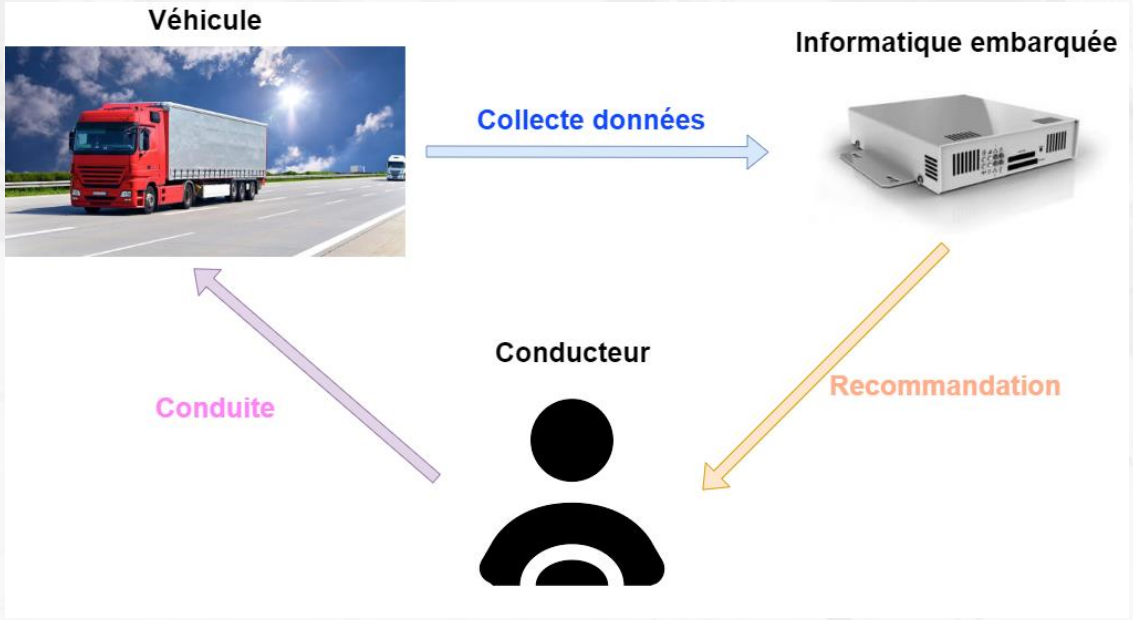
Contexte :

- STRADA souhaite capitaliser sur cette masse de données.
- Un nombre important de clients ont manifesté l'intérêt de mettre en place une solution pour réduire la consommation de carburant et l'émission de CO2.
- Il existe des solutions d'aide à la conduite mais avec un principe de fonctionnement basé sur les recommandations à posteriori.
- ***STRADAlert*** : un projet pour mettre en place une solution d'aide à la conduite (sur l'aspect éco-conduite) en temps réel.

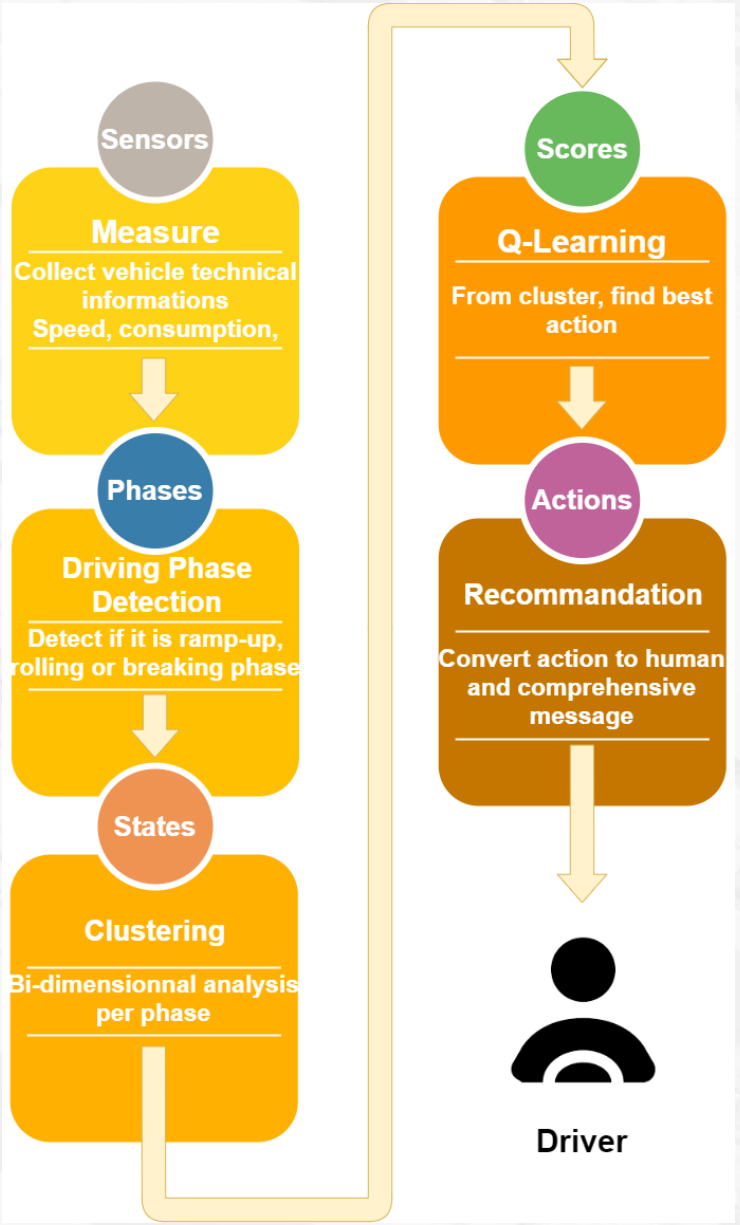
STRADAlert (RPI PROTOTYPE) :

Principe de fonctionnement :

- La solution devra s'abonner sur les données de conduite sur le véhicule (bus CAN, accéléromètre, …).
- Les données seront traitées et injectées dans un ensemble d'algorithmes (ML, RL).
- La solution devra fournir en sortie, des recommandations adéquates au conducteur en temps réel pour améliorer l'aspect éco-conduite de sa conduite.



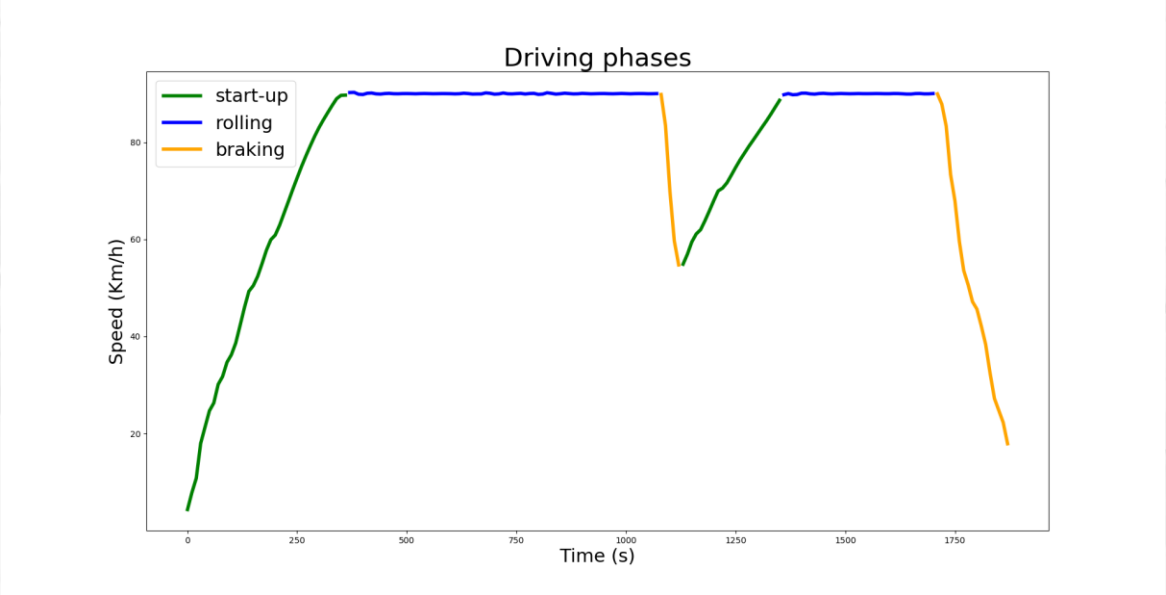
Workflow :



Modélisation de l'environnement

Phases de conduite :

- *Start-up Phase* : Cette phase correspond à la détection d'une forte accélération sur une fenêtre de temps assez courte.
- *Rolling Phase* : Cette phase correspond au maintien d'une vitesse constante (pas d'accélération ni de décélération).
- *Braking Phase* : Cette phase correspond à la détection d'une forte décélération sur une fenêtre de temps assez courte.

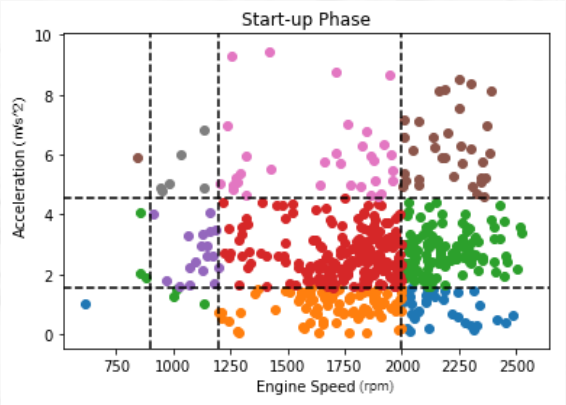


Jeu de données :

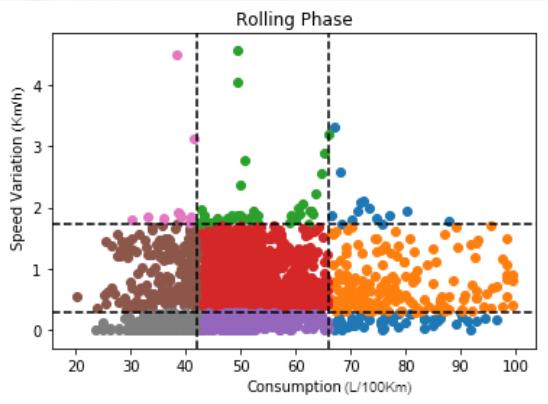
On a exploité 6 Features des données récoltées : ***Accélération***, ***Régime moteur***, ***Variation de vitesse***, ***Consommation de carburant***, ***Déclaration***, ***Pourcentage de freinage***.

Espace de représentation des phases :

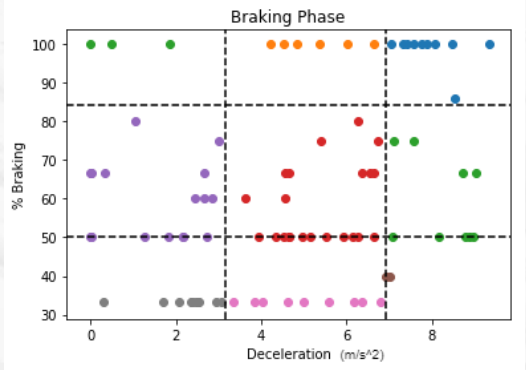
Start-up Phase



Rolling Phase

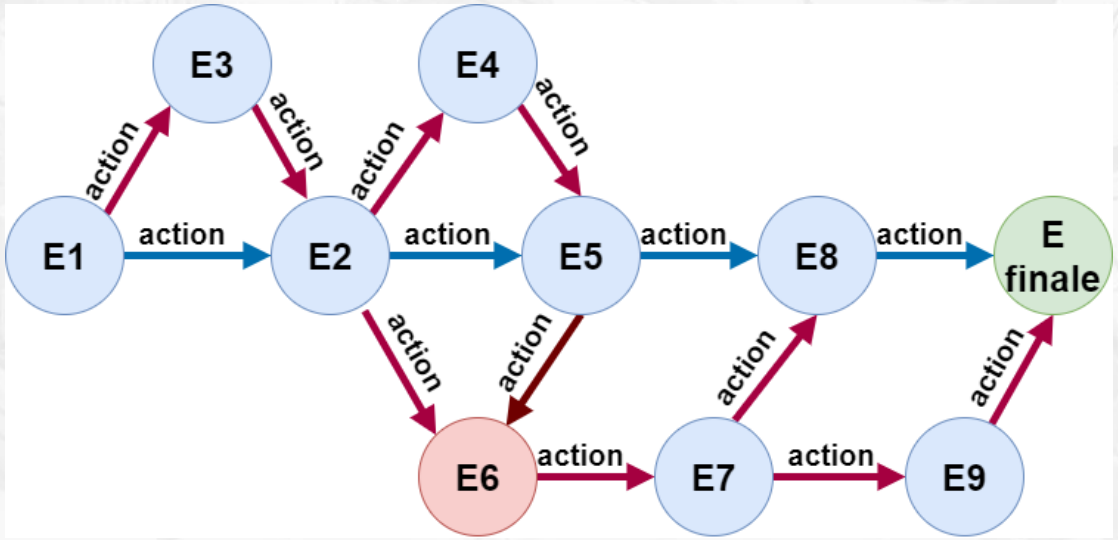


Braking Phase



Reinforcement Learning :

Un agent qui devra apprendre les actions à appliquer à partir d'expériences. Cet apprentissage s'effectue en optimisant une récompense quantitative itérativement. RL est une classe de solutions pour résoudre Les systèmes décisionnels de Markov.



Objectifs :

- Choisir le chemin le plus court.
- Cumuler le maximum de récompenses.
- Privilégier les récompenses à court terme que celles à long terme.

Q-Learning :

Le Q-Learning est régi par le Markov Decision Processes (MDPs). Le Q-Learning consiste à déterminer une fonction (une table) $Q(s,a)$ qui prend deux paramètres :

- s : l'état du système.
- a : l'action que l'on veut effectuer.

État	↑	→	↓	←
1	0.85	0.10	0.35	0.56
2	0.10	0.65	0.55	0.75
…				
N	0.63	0.55	0.98	-0.56

Actions

$$Q(s_t,a_t)_{new} = Q(s_t,a_t) + \alpha \left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]$$

L'espace des actions et récompenses :

Start-up Phase

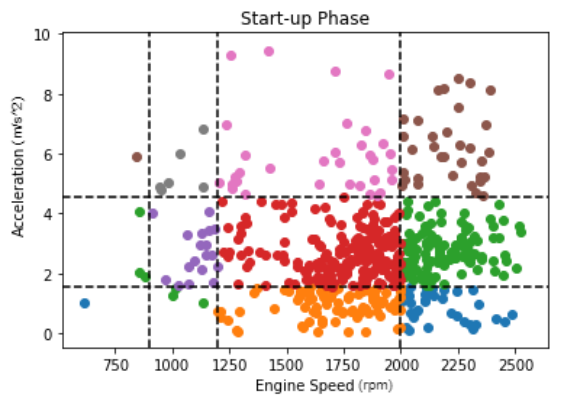
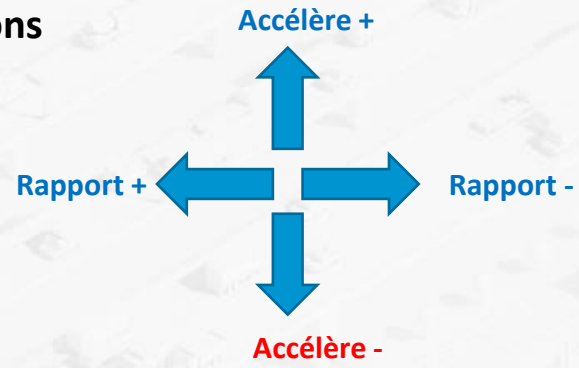


Table des Scores

6	8	10	2
2	3	4	1
-1	0	2	-1

Actions



Rolling Phase

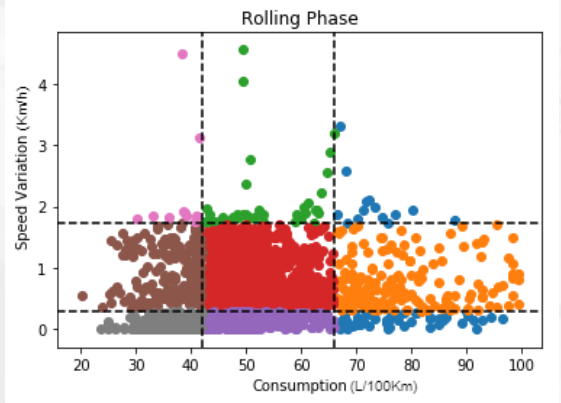
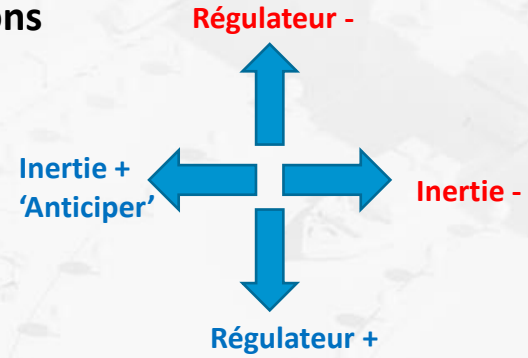


Table des Scores

3	2	-1
6	4	0
10	8	1

Actions



Braking Phase

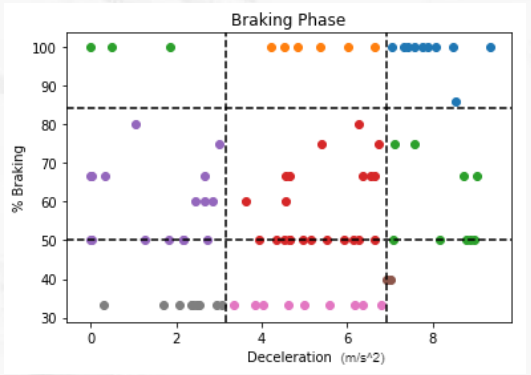
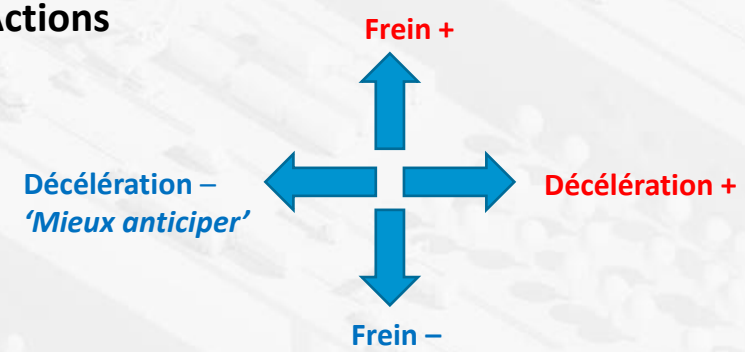


Table des Scores

2	0	-1
4	3	2
10	8	6

Actions

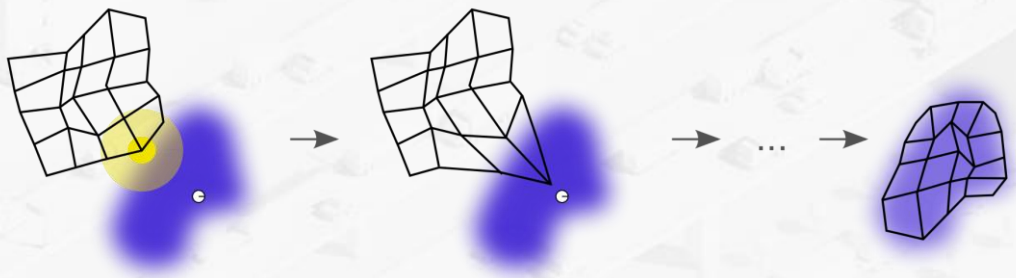


Identification des états :

L'objectif est d'identifier les états dans chacun des espaces de représentation, Pour cela, nous avons appliqué l'algorithme Kohonen 2D (SOM).

Kohonen 2D :

- Initialisation aléatoire des neurones.
- Loop [0: n_iter] :
  - Choix aléatoire d'un échantillon.
  - Calculer la distance entre l'échantillon et tous les neurones.
  - Choisir le neurone gagnant (le plus proche).
  - Corriger les positions des neurones en fonction du neurone gagnant.

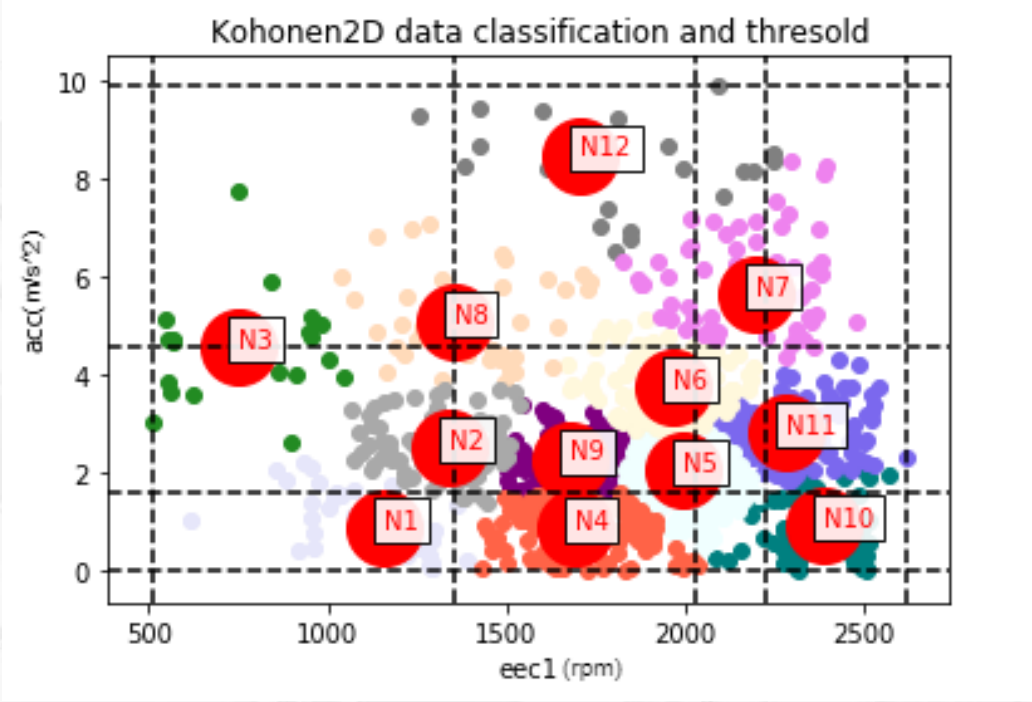


Calcul des seuils :

Le but est de calculer les seuils (limites des états) en fonction des positions des neurones :

- Loop sur tous les groupes :
  - Récupérer les ***pt_North***, ***pt_South***, ***pt_Est***, ***pt_West.***
  - Enregistrer ***pt_North***, ***pt_South*** dans ***limits_y*** et ***pt_Est***, ***pt_West dans limits_x***.
  - Appliquer un calcul de quantile sur les vecteurs ***limits_x*** et ***limits_y*** en fonction de la taille de la table de Scores.

Résultats Start-up Phase



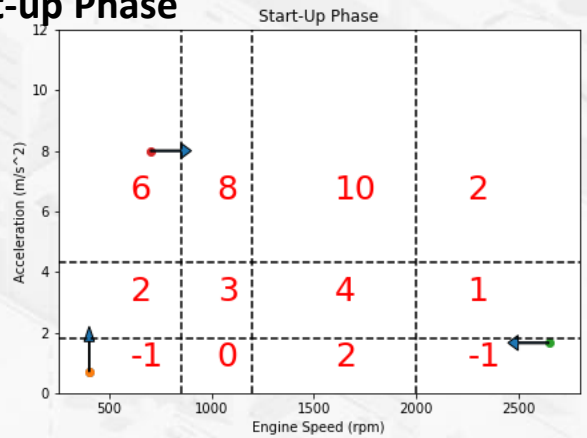
Simulation et entrainement :

- Nécessité d’avoir un contrôle complet des données.
- Enregistrer les données de conduite.
- Exploiter une seconde source de données : ***Euro Truck Simulator 2***.

Euro Truck Simulator 2

Résultats :

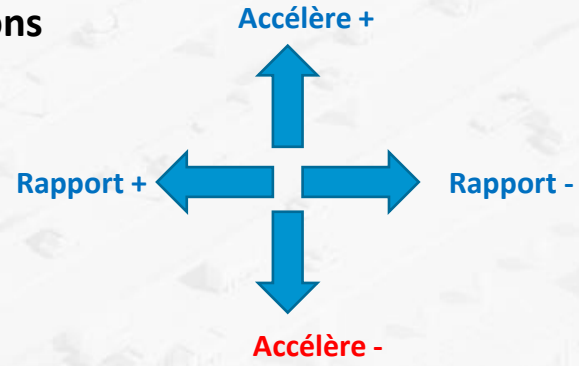
Start-up Phase



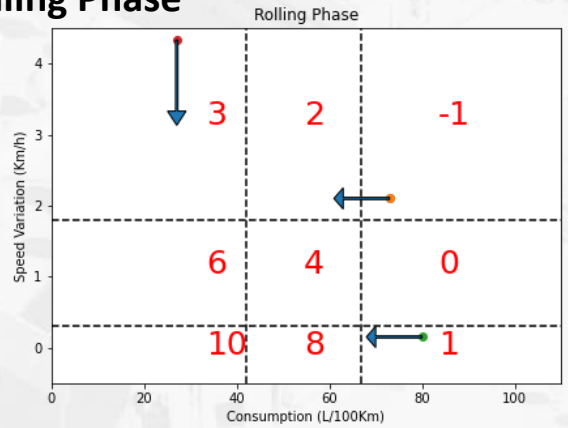
Coordonnées points

Dataset point	Recommendation	Score
engine speed = 400 rpm, acceleration = 0.7 m/s^2	*more acceleration*	-1
engine speed = 2650 rpm, acceleration = 1.655 m/s^2	*next gear*	-1
engine speed = 700 rpm, acceleration = 8 m/s^2	*previous gear*	+6

Actions



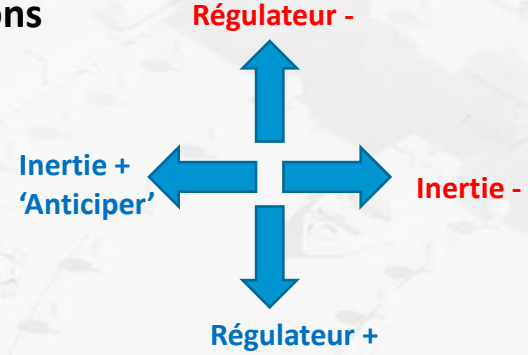
Rolling Phase



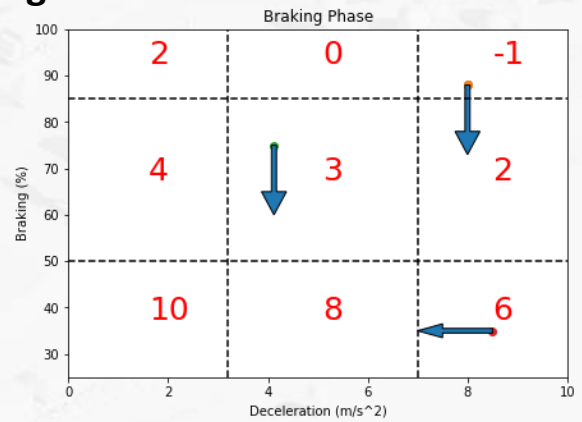
Coordonnées points

Dataset point	Recommendation	Score
fuel consumption = 73 $L/100Km$, speed variation = 2.1 Km/h	*use more inertia*	-1
fuel consumption = 80 $L/100Km$, speed variation = 0.15 Km/h	*use more inertia*	+1
fuel consumption = 27.1 $L/100Km$, speed variation = 4.33 Km/h	*use more regulator*	+3

Actions



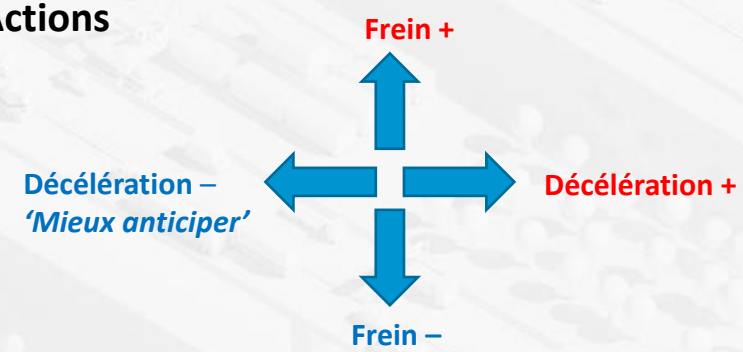
Braking Phase



Coordonnées points

Dataset point	Recommendation	Score
deceleration = 8 m/s^2, braking = 88 %	*use less brake*	-1
deceleration = 4.12 m/s^2, braking = 75 %	*use less brake*	+3
deceleration = 8.5 m/s^2, braking = 35 %	*less deceleration*	+6

Actions



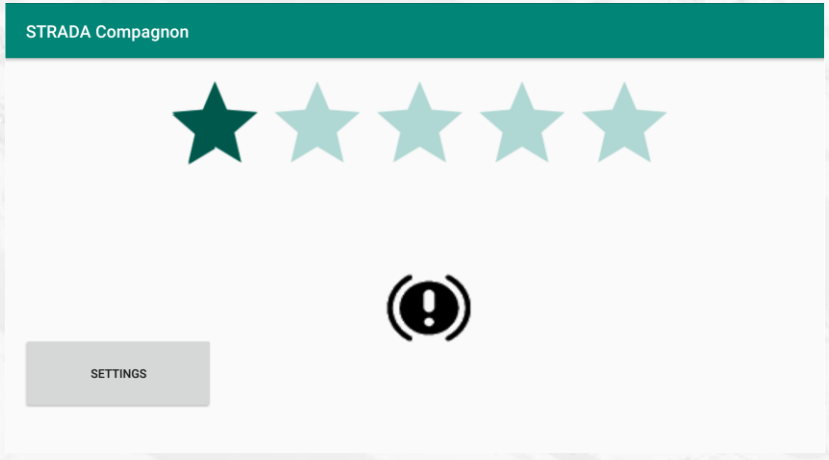
Résultats :

Illustrations de recommandations

More acceleration

More regulator

Next gear



Less break

Bilan :

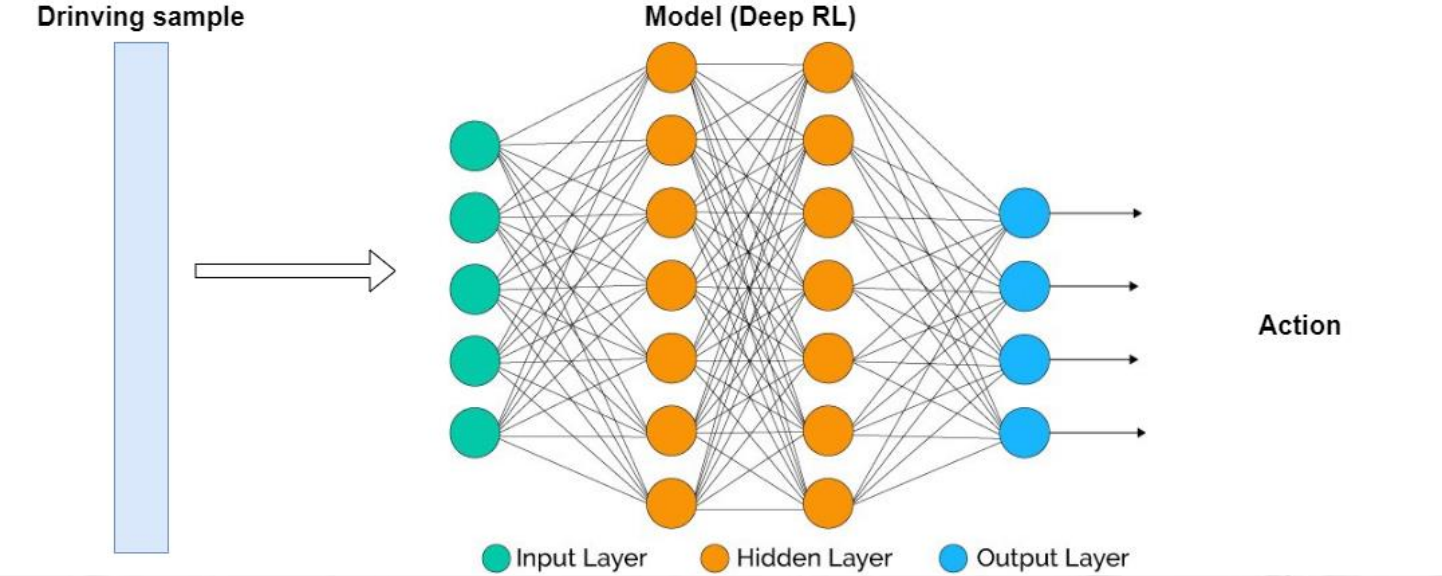
- La solution a la capacité de produire des recommandations adéquates en fonction de la conduite du conducteur.
- La difficulté réside dans l'identification des états.
- Trouver une meilleure solution qui nous permettra de nous défaire du besoin d'identifier les états.

Solution Deep RL :

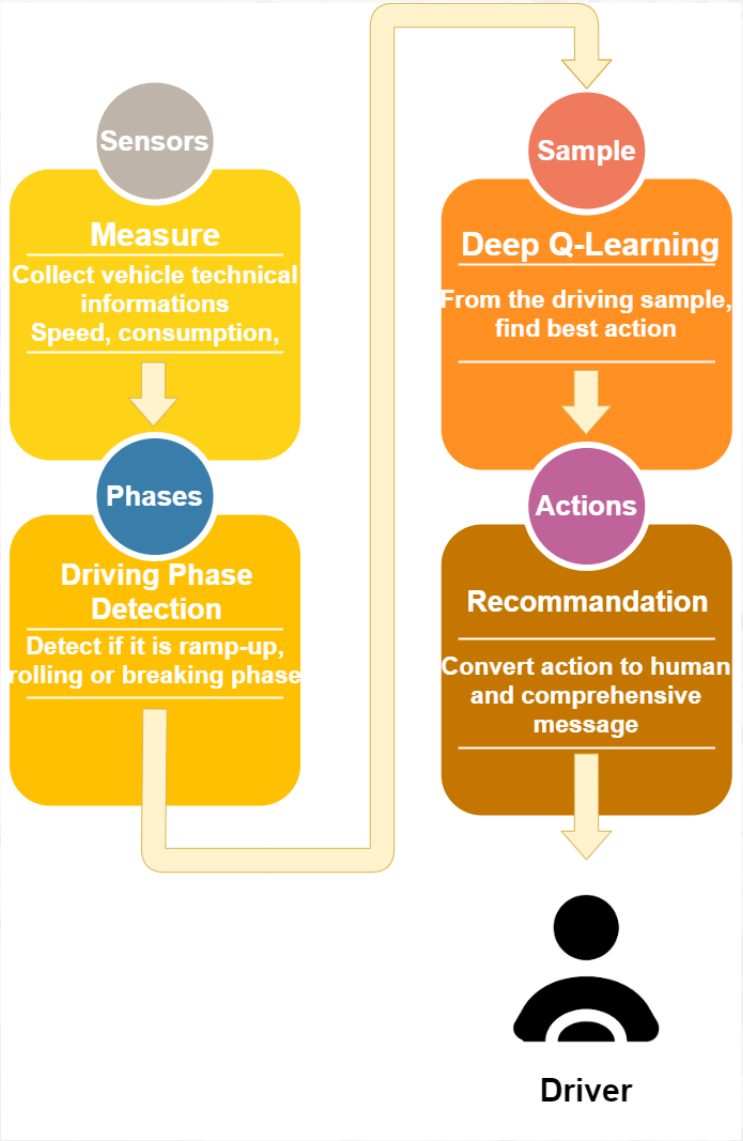
Le but de cette nouvelle solution est d'utiliser un algorithme Deep RL :

- Permettre de nous défaire de la partie identification des états.
- Un processus plus automatisé.

Le modèle devra de lui-même, à partir d'un échantillon donné, identifier la meilleure action sans passer par l'identification de l'état.

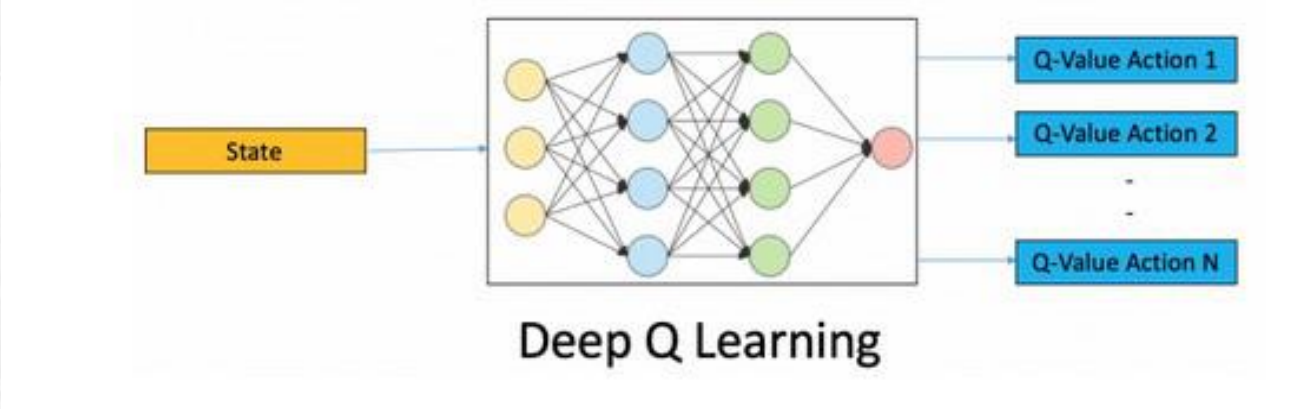


Workflow :



Deep Q-Learning :

Nous avons utilisé l'algorithme Deep Q-Learning.

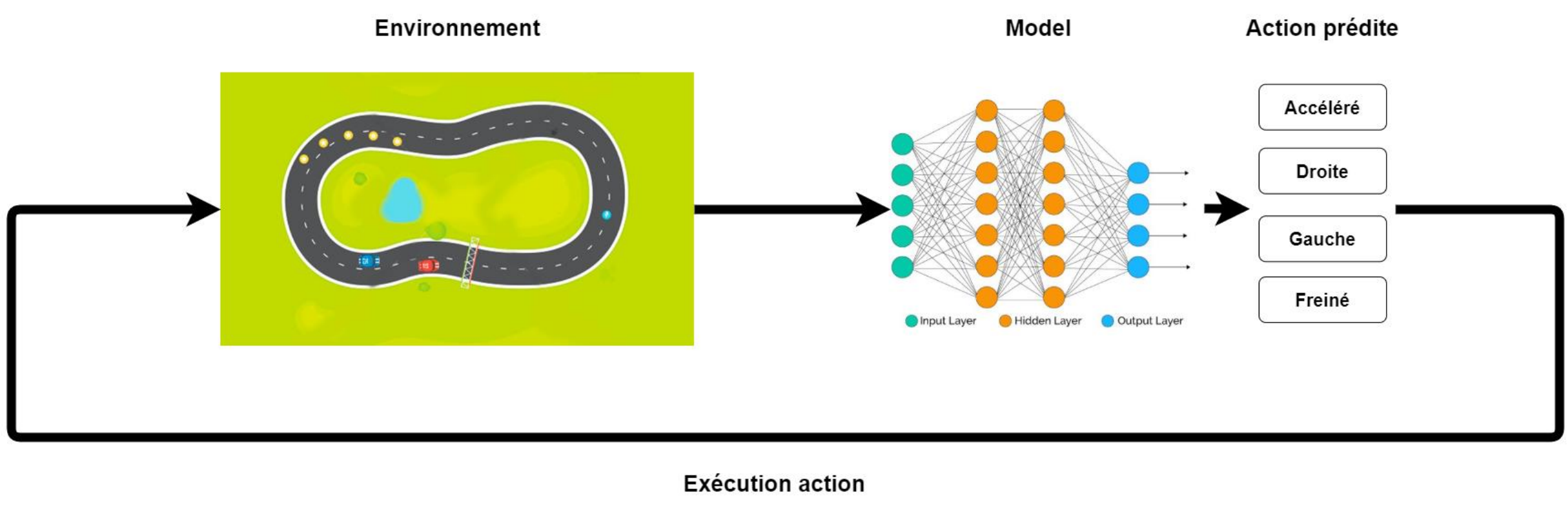


Algorithme :

- Loop :
  - Choisir l'action ***a*** pour l'état ***s*** (meilleure Q-Value).
  - Exécuter l'action ***a*** -> passage vers l'état ***s'***.
  - Récupérer le Reward ***r***.
  - **Update des poids des neurones**.

Deep Q-Learning

Cas classique

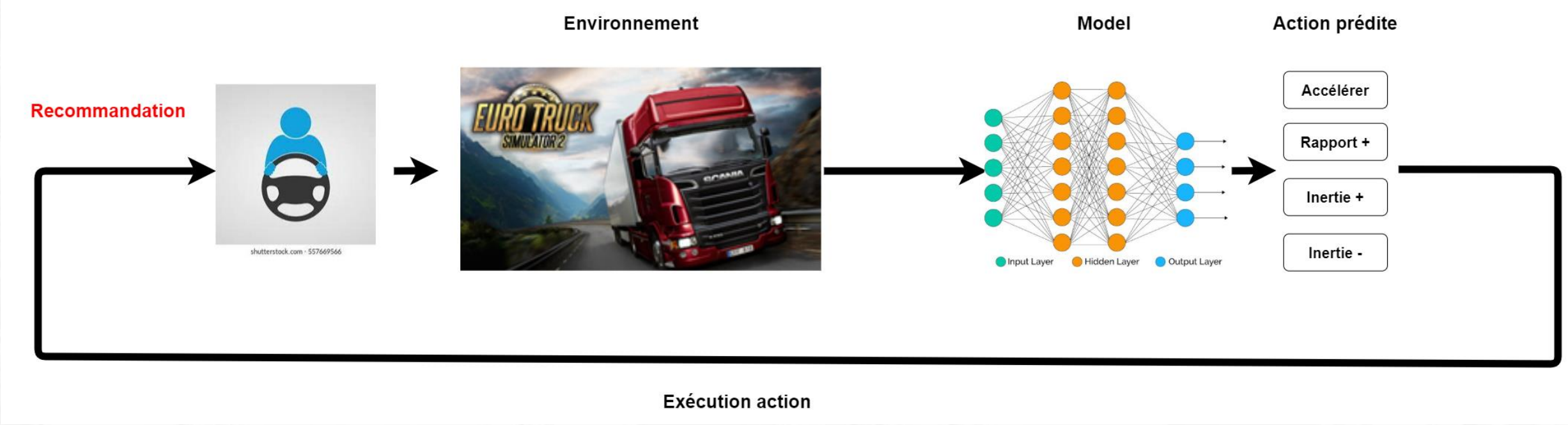


Phase d'apprentissage :

- L'action prédite impacte complètement l'environnement.
- Le modèle a un contrôle complet sur la voiture.

Deep Q-Learning

Problématique

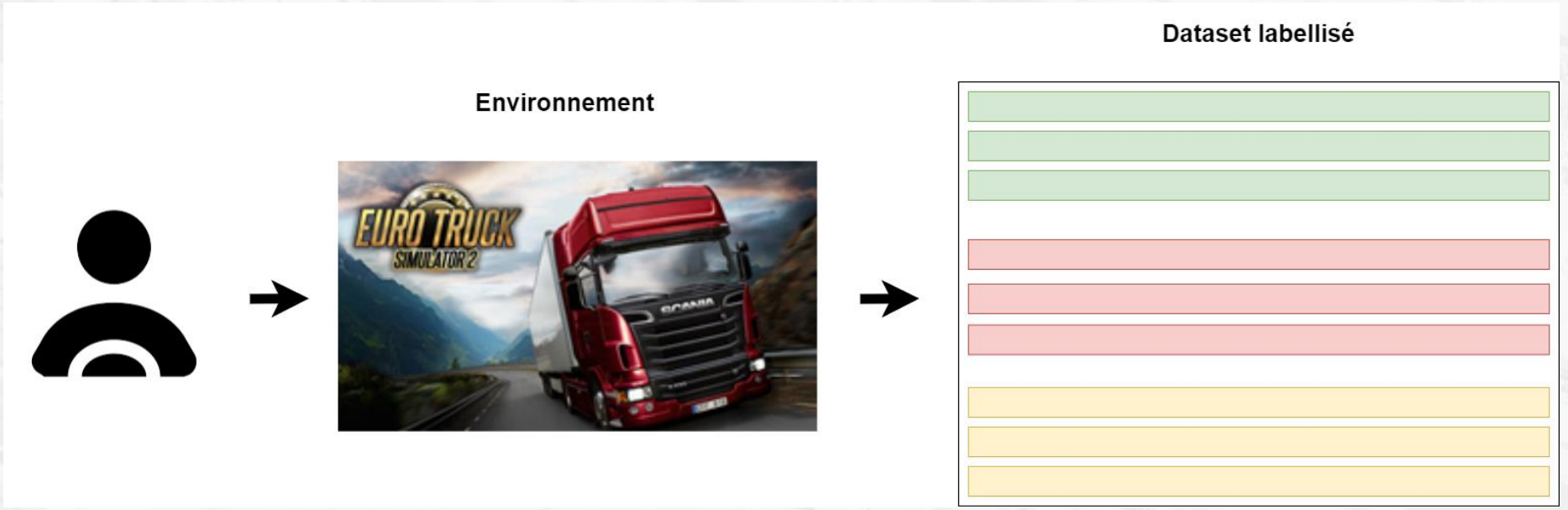


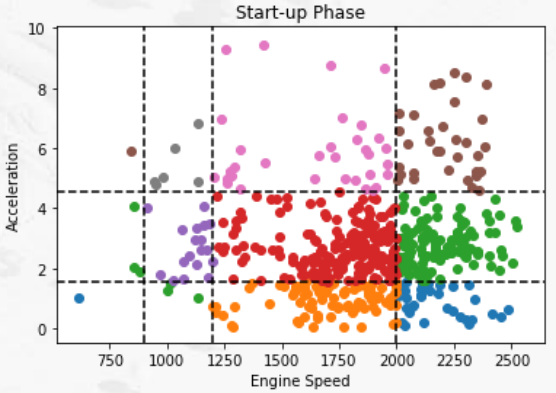
Phase d'apprentissage :

- L'action prédite est une recommandation.
- L'action ne sera pas forcément exécutée par l'agent.

Deep Q-Learning

Proposition





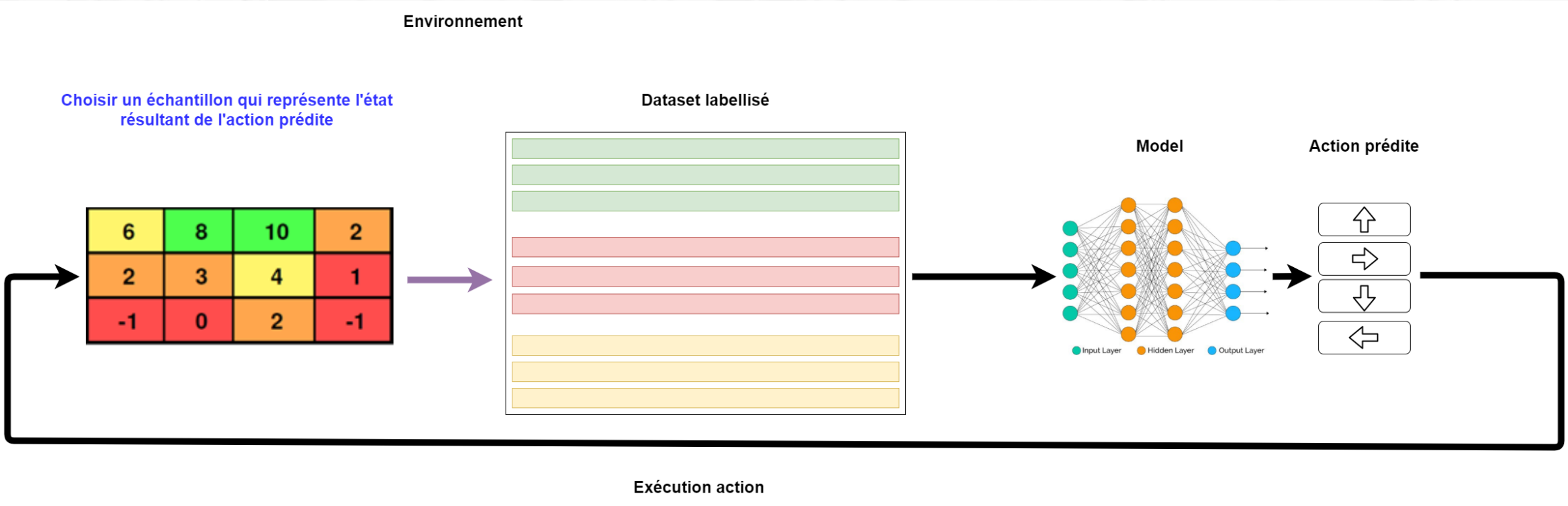
6	8	10	2
2	3	4	1
-1	0	2	-1

Création d'un Dataset labellisé :

- Un Dataset qui contient un nombre d'échantillons suffisant pour chaque état.

Deep Q-Learning

Proposition



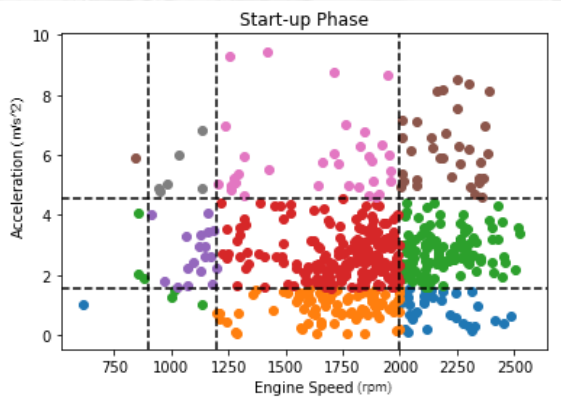
Phase d'apprentissage :

- Injecter un premier état aléatoirement (vecteur de données).
- Le modèle propose une action.
- On choisit l'état suivant en fonction de l'action.
- On injecte un nouveau vecteur de données qui correspond au nouvel état.
- Loop.

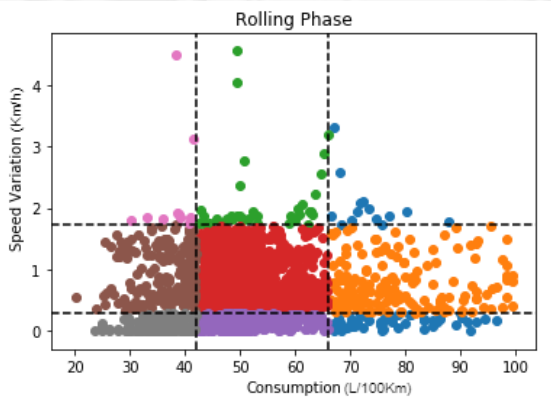
Deep Q-Learning

Résultats

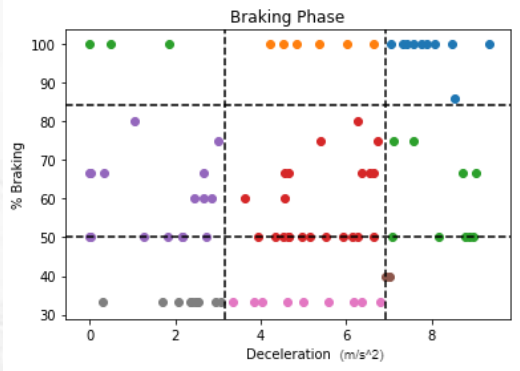
Start-up Phase



Rolling Phase



Braking Phase



- Il y a un fort déséquilibre entre les différentes classes (états).
- Des approches basées sur la pondération des classes, le sur-échantillonnage ou le sous-échantillonnage n'ont pas permis de régler le problème.

Conclusion

- Nous avons proposé ici une solution d'aide à la conduite en temps réel.
- La solution a la capacité de produire des recommandations adéquates en fonction de la conduite du conducteur.
- La solution basée sur du Deep RL (pour se défaire de la partie d'identification des états) n'est pas adaptée au cas d'étude :
  - Pas de lien direct avec l'environnement.
  - Données labellisées avec un déséquilibre des classes très important.
- Notre solution présente néanmoins quelques lacunes :
  - La difficulté de proposer des recommandations cohérentes pour des conduites sur des topologies de routes spécifiques (en montée, en descente, …).
  - L'incapacité de pondérer les recommandations en fonction des marchandises transportées par le véhicule.
- Propositions :
  - Inclure les informations sur les topologies de routes, marchandises, poids marchandises, … Et travailler sur des espaces de représentations avec des dimensions > 2.
  - Modélisation de phases de conduites supplémentaires.