RAPPORT D'ACTIVITÉ

Projet de fin de module RL

Agent de Conduite Autonome Basé sur

l'Apprentissage par Renforcement

Self-Driving with CARLA

Master 2 Intelligence Artificielle

Année Universitaire 2025-2026

Réalisé par :

Théo CONDAMIN

Louis LAMBERT

Paul BERDIER

Table des matières

1. Introduction

1.1 Contexte et Problématique

La conduite autonome représente l'un des défis majeurs de l'intelligence artificielle appliquée. Dans le cadre de notre projet de fin de module renforcement learning de Master 2, nous avons développé un agent de conduite autonome capable de se déplacer de manière indépendante, d'adapter sa vitesse en fonction de la densité du trafic et de gérer efficacement l'environnement de simulation CARLA.

L'apprentissage par renforcement (Reinforcement Learning) offre une alternative prometteuse aux approches traditionnelles basées sur des règles prédéfinies. Cette méthode permet aux véhicules d'apprendre des stratégies de conduite optimales par l'expérience directe dans un environnement simulé.

1.2 Objectifs du Projet

Objectif principal : Développer un agent capable de conduire de manière autonome dans l'environnement CARLA.

Objectifs secondaires :

- Maintenir le véhicule centré dans sa voie (lane keeping)
- Éviter les collisions avec les obstacles
- Maintenir une vitesse appropriée selon le contexte
- Progresser vers une destination définie
- Assurer une conduite stable et prévisible

1.3 Choix Technologiques

Composant	Technologie	Justification
Simulateur	CARLA 0.9.13+	Open-source, réaliste, API Python complète
Framework RL	PyTorch 2.0+	Flexibilité, support GPU, grande communauté
Algorithme	PPO	Stabilité, efficacité d'échantillonnage, simplicité
Environnement	Gymnasium	Standard dans la communauté RL

2. Hypothèses et Contraintes

Dès le début du projet, nous avons identifié plusieurs contraintes majeures qui ont orienté nos choix techniques et méthodologiques.

2.1 Hypothèses Formulées

Hypothèse 1 : Limitations des caméras RGB

L'utilisation de caméras RGB serait trop gourmande en ressources computationnelles. Cette hypothèse nous a conduits à privilégier des capteurs LIDAR plus légers et à abandonner l'approche vision par ordinateur pour la V0.

Hypothèse 2 : Lourdeur de l'environnement CARLA

L'environnement CARLA étant particulièrement exigeant en ressources, nous avons anticipé qu'il serait impossible d'effectuer un grand nombre d'épisodes d'entraînement sur nos machines personnelles.

Hypothèse 3 : Stratégie de convergence rapide

Face à ces limitations, nous avons choisi de focaliser nos efforts sur la qualité de la fonction de récompense et la sélection du modèle pour apprendre avec le minimum d'épisodes possible.

2.2 Ressources Disponibles

Limitations matérielles :

- Machines personnelles avec ressources CPU/GPU limitées
- Pas d'accès à des clusters de calcul haute performance
- Temps d'entraînement contraint par la disponibilité des machines

Limitations temporelles :

- Délai de livraison du projet limité
- Temps d'apprentissage de CARLA et des algorithmes RL
- Nécessité de tester de multiples approches

3. Environnement CARLA

3.1 Présentation de CARLA

CARLA (Car Learning to Act) est un simulateur open-source de conduite urbaine virtuel développé pour la recherche en conduite autonome. Il offre un environnement réaliste avec gestion de la physique, des conditions météorologiques et du trafic.

Au cours du projet, nous avons utilisé plusieurs outils et capteurs pour collecter des observations de l'environnement et permettre à notre agent d'interagir efficacement.

3.2 Capteurs et Observations

Configuration des capteurs :

- **Capteur de position :** localisation du véhicule dans l'environnement
- **Capteur de vitesse :** vitesse actuelle du véhicule en m/s
- **Capteur de collision :** détection et intensité des collisions
- **LIDAR :** 16 distances minimum + 16 distances moyennes par secteur angulaire
- **Waypoints :** points de navigation sur la carte routière
- **Caméra RGB :** testée mais abandonnée pour la V0 (trop gourmande)

3.2.1 Observations Finales (37 dimensions)

Après plusieurs itérations, nous avons convergé vers un vecteur d'observation de 37 dimensions :

- **16 distances minimum** (LIDAR, valeurs normalisées dans [0, 50m])
- **16 distances moyennes** (LIDAR, valeurs normalisées)
- **1 collision** (intensité de la collision)
- **1 position** (distance au point de spawn)
- **1 vitesse** (vitesse actuelle normalisée)
- **1 lane offset** (décalage latéral par rapport au centre de la voie)
- **1 lane angle** (angle du véhicule par rapport à la direction de la voie)

3.3 Espace d'Actions

Notre agent dispose de trois actions continues pour contrôler le véhicule :

- **Direction (steering) :** valeur continue de -1 (gauche) à 1 (droite)
- **Accélération (throttle) :** valeur continue de 0 (aucune) à 1 (maximale)
- **Freinage (brake) :** valeur continue de 0 (aucun) à 1 (freinage maximal)

Ces actions sont échantillonnées depuis une distribution gaussienne multivariée, puis transformées via une fonction sigmoïde pour garantir qu'elles restent dans les plages attendues.

4. Composition de la Récompense

La fonction de récompense constitue le cœur de notre système d'apprentissage. Elle a été conçue pour équilibrer plusieurs objectifs parfois contradictoires : progresser rapidement, rester centré dans la voie, éviter les collisions et maintenir une conduite stable.

4.1 Récompenses à Chaque Step

Base Reward (+0.01)

Favorise la survie de l'agent en lui donnant une récompense constante pour chaque step effectué.

Lane Keeping Reward (+0.25 à +0.55)

Favorise le maintien du véhicule au centre de sa voie avec un minimum d'écart angulaire. Cette récompense est progressive :

- **Excellent :** offset ≤ 0.1 ET angle ≤ 0.05 → +0.25 à +0.55 (avec bonus de série)
- **Good :** offset ≤ 0.3 ET angle ≤ 0.15 → +0.1 à +0.2
- **Acceptable :** offset ≤ 0.7 ET angle ≤ 0.4 → +0.025
- **Mauvais :** au-delà → pénalité quadratique basée sur l'offset et l'angle

Consistency Bonus (+0.05 à +0.2)

Favorise une conduite stable et prolongée dans la voie :

- 50+ steps consécutifs de bon maintien → +0.2
- 20-49 steps → +0.1
- 10-19 steps → +0.05

Exploration Reward (+0.1 maximum)

Favorise la progression en récompensant la distance parcourue depuis le point de spawn, avec un plafond pour éviter une exploration anarchique.

4.2 Pénalités Critiques

Collision (-500)

Pénalise fortement les collisions avec les obstacles pour dissuader l'agent de comportements dangereux.

Immobility Penalty (-0.001 par step)

Pénalise l'immobilité excessive pour éviter que l'agent ne reste statique pour maximiser sa survie.

Off-Road Penalty (variable)

Pénalité croissante (exponentielle) pour le nombre de steps passés trop loin ou trop désaxé de la voie de circulation.

Off-Road Termination (-250)

Pénalise fortement et termine l'épisode si le véhicule sort complètement de la route (>30 steps off-road ou offset >0.95).

4.3 Évolution et Stabilisation

Un système de clipping a été mis en place pour stabiliser la récompense à chaque step et éviter les valeurs extrêmes qui pourraient déstabiliser l'apprentissage. Les récompenses sont limitées dans l'intervalle [-150, 150].

Le graphique d'évolution de la récompense montre l'entraînement d'un modèle sélectionné pour sa convergence rapide après les 10 premiers épisodes, ce qui confirme l'efficacité de notre fonction de récompense équilibrée.

5. Évolution du Projet et du Modèle

Le développement de notre agent de conduite autonome s'est déroulé en plusieurs phases distinctes, chacune apportant son lot d'enseignements et d'optimisations.

5.1 Phase de Découverte de CARLA

Objectifs de cette phase :

- Étude approfondie de l'environnement CARLA
- Maîtrise du contrôle de CARLA et du système de Blueprint
- Spawn de véhicules, de caméras et contrôle des acteurs
- Optimisation pour faire tourner l'environnement sur nos machines

Difficultés rencontrées :

- Configuration complexe de l'environnement
- Instabilité du serveur CARLA sur machines personnelles
- Temps de chargement importants entre chaque épisode

5.2 Premiers Modèles (DQN)

Travaux réalisés :

- Structuration complète du code de l'environnement
- Ajout de multiples capteurs sur le véhicule (LIDAR, caméra RGB, etc.)
- Implémentation d'un DQN avec Replay Buffer et encodeur pour LIDAR
- Mise en place de boucles d'entraînement et graphiques pour suivre l'évolution de la reward
- Tests et itérations multiples

Limitations identifiées :

- Difficultés à trouver de la performance avec DQN
- Modèle vu en cours insuffisant dans notre situation pour la V0
- Entraînement DQN insupportable par nos machines (temps + ressources)
- Nécessité de chercher un modèle plus efficient

5.3 Modèle Plus Efficient (Policy Gradient)

Recherche et sélection :

- Recherche approfondie d'un nouveau modèle et approche
- Sélection d'algorithmes policy gradient (TD3, SAC, PPO)
- Comparaison des performances et de la complexité d'implémentation

Optimisations techniques :

- Suppression de l'encodeur du LIDAR et de la caméra RGB

- Simplification du traitement LIDAR (binning angulaire direct)
- Intégration de lane_offset et lane_angle pour améliorer le centrage
- Optimisation de la charge GPU et CPU

Défis persistants :

- Itérations multiples pour rechercher l'efficience
- Difficultés à obtenir une convergence stable
- Perte de motivation due à l'éloignement du livrable final
- Convergence des modèles résultant plus de la chance que de l'optimisation

5.4 Dernières Optimisations

Ajustements finaux :

- Modification légère des poids de la reward pour équilibrage
- Lancement d'entraînements en boucle pour maximiser les chances de succès
- Tests de la V1 du modèle

Problème technique critique :

Plus le modèle devenait performant, plus sa vitesse et l'intensité des collisions augmentaient durant l'entraînement. Cette situation a conduit à l'arrêt forcé de l'entraînement car les machines bloquaient à cause de la charge de calcul excessive.

6. Choix Final du Modèle : PPO

6.1 Justification du Choix

Après avoir testé plusieurs approches (DQN, TD3, SAC), nous avons finalement sélectionné l'algorithme Proximal Policy Optimization (PPO) pour les raisons suivantes :

- **Modèle de policy gradient :** plus pertinent car il produit des actions en valeurs continues sans discrétisation, ce qui permet une précision accrue et une meilleure exploration
- **Modèle on-policy :** réduit la divergence de l'apprentissage en n'utilisant que des trajectoires récentes
- **Très stable :** grâce au mécanisme de clipping qui empêche les mises à jour trop agressives
- **Input only :** non compatible avec des images brutes mais parfait pour nos observations vectorielles
- **Convergence rapide :** nécessite bien moins d'inputs et d'épisodes pour converger que DQN
- **Facilité d'implémentation :** plus simple à implémenter que SAC ou TD3
- **Coût en ressources :** moins coûteux en ressources computationnelles

6.2 Architecture du Réseau de Neurones

Notre implémentation PPO utilise une architecture Actor-Critic avec les caractéristiques suivantes :

Structure du réseau :

- **Couche d'entrée :** 37 neurones (vecteur d'observation)
- **Couche cachée 1 :** 256 neurones avec activation ReLU
- **Couche cachée 2 :** 256 neurones avec activation ReLU
- **Tête Actor :** 3 sorties (steering, throttle, brake)
- **Tête Critic :** 1 sortie (estimation de la valeur de l'état)

6.3 Hyperparamètres

Paramètre	Valeur
Learning Rate	3e-4
Clipping PPO (ε)	0.2
Époques par Update	10
Discount Factor (γ)	0.99
GAE Parameter (λ)	0.95
Value Coefficient (c_1)	0.5
Entropy Coefficient (c_2)	0.05
Max Steps par Épisode	1000

7. Résultats

7.1 Performances de l'Agent

Notre agent a réussi à apprendre certains comportements de base :

- **Suivre la route :** capacité démontrée à maintenir le véhicule sur la chaussée
- **Maintien dans la voie :** ne change pas de voie de manière anarchique
- **Respect du système RL :** utilise correctement les vecteurs d'observations, l'ensemble d'actions et la fonction de récompense

7.2 Limitations Identifiées

Malgré ces succès, plusieurs limitations persistent :

- **Gestion des virages :** l'agent ne suit pas correctement les virages serrés
- **Adaptation de la vitesse :** ne ralentit pas correctement face aux obstacles
- **Changement de voie :** incapacité à effectuer des changements de voie délibérés

7.3 Analyse des Résultats

Les résultats obtenus sont objectivement insuffisants par rapport aux ambitions initiales du projet. Cependant, au vue des ressources allouées (temps limité, matériel personnel, pas d'accès à des clusters), le travail effectué est honorable et démontre une compréhension solide des principes de l'apprentissage par renforcement.

8. Contraintes et Limitations

Le projet a été significativement impacté par plusieurs types de limitations qui ont restreint l'amplitude de nos ambitions et la qualité des résultats finaux.

8.1 Manque de Temps

- Durée limitée du projet académique (un semestre)
- Courbe d'apprentissage importante pour CARLA et les algorithmes RL
- Temps d'entraînement très long pour chaque modèle
- Nombreuses itérations nécessaires pour ajuster les hyperparamètres

8.2 Manque de CPU & GPU

- Utilisation uniquement de machines personnelles
- Pas d'accès à des clusters de calcul haute performance
- Impossibilité de lancer plusieurs entraînements en parallèle
- Blocages fréquents des machines lors d'entraînements intensifs
- Limitation du nombre d'épisodes d'entraînement possibles

8.3 Manque de Compétences Initiales

- Découverte de CARLA pendant le projet
- Apprentissage des algorithmes RL avancés en parallèle
- Manque d'expérience en design de fonctions de récompense
- Nécessité d'acquérir des compétences en debugging de modèles RL

8.4 Impact sur le Projet

Ces limitations combinées ont forcé l'équipe à adopter une stratégie pragmatique : plutôt que de viser l'excellence technique absolue, nous avons concentré nos efforts sur la mise en place d'une architecture fonctionnelle et la compréhension approfondie des mécanismes d'apprentissage par renforcement.

9. Conclusion

9.1 Synthèse du Projet

Ce projet de conduite autonome basé sur l'apprentissage par renforcement s'est révélé être un défi technique et pédagogique majeur. Nous avons développé un agent capable d'évoluer dans l'environnement CARLA en utilisant l'algorithme PPO, avec une fonction de récompense multi-objectifs soigneusement conçue.

Le projet était intéressant mais sans doute trop ambitieux compte tenu des ressources disponibles. Les résultats obtenus, bien qu'objectivement insuffisants par rapport à nos objectifs initiaux, représentent un travail honorable eu égard aux contraintes matérielles, temporelles et de compétences rencontrées.

9.2 Acquis Techniques

Compétences développées :

- Maîtrise du simulateur CARLA et de son API Python
- Implémentation d'algorithmes d'apprentissage par renforcement (DQN, PPO)
- Design et calibration de fonctions de récompense complexes
- Gestion de l'environnement Gymnasium pour RL
- Optimisation de pipelines d'entraînement sous contraintes de ressources
- Utilisation de PyTorch pour le deep learning

9.3 Leçons Apprises

- **Importance de l'infrastructure :** les ressources computationnelles sont critiques pour le RL
- **Design de reward :** une fonction de récompense bien conçue est plus importante qu'un modèle complexe
- **Choix d'algorithme :** PPO offre un excellent compromis stabilité/efficacité pour nos contraintes
- **Itération et patience :** le RL nécessite de nombreux essais et ajustements
- **Réalisme des objectifs :** adapter l'ambition du projet aux ressources disponibles

9.4 Perspectives d'Amélioration

Court terme :

- Correction du bug sigmoid/atanh identifié dans le code
- Implémentation de curriculum learning pour progression graduelle
- Entraînement sur davantage d'épisodes avec accès à du matériel plus puissant

Moyen terme :

- Intégration de caméras RGB avec encodeur CNN
- Entraînement sur plusieurs cartes CARLA pour généralisation

- Ajout de trafic dynamique et d'obstacles mobiles

Long terme :

- Implémentation de Safe RL avec contraintes formelles
- Scénarios multi-agents avec interaction entre véhicules
- Transfer learning vers environnements réels

9.5 Mot de Fin

Malgré les défis rencontrés et les résultats en deçà de nos espérances, ce projet nous a permis d'acquérir une expérience pratique précieuse en apprentissage par renforcement profond. Nous avons appris à naviguer entre ambitions théoriques et contraintes pratiques, une compétence essentielle pour tout projet de recherche appliquée.

L'équipe remercie l'encadrement académique pour l'opportunité de travailler sur ce sujet passionnant et espère que ce travail pourra servir de base à de futurs projets plus ambitieux bénéficiant de ressources accrues.

Annexes

Annexe A : Architecture Système Complète

Le système se compose de trois modules principaux :

1. Module Environnement (simulation_V6.py)

- Interface avec le serveur CARLA
- Gestion du cycle de vie du véhicule
- Collecte des données capteurs
- Calcul des récompenses
- Détection des conditions de terminaison

2. Module Modèle (model_PPO_V6.py)

- Implémentation de l'architecture Actor-Critic
- Distribution gaussienne pour échantillonnage des actions
- Calcul des log-probabilités pour le gradient
- Optimisation via Adam

3. Module Entraînement (main_V6.py)

- Connexion au serveur CARLA
- Chargement/Création du modèle
- Rollout : collection de trajectoires
- Calcul GAE : avantages et returns
- Update PPO : optimisation sur 10 époques
- Sauvegarde régulière de checkpoints
- Génération de graphiques de convergence

Annexe B : Configuration CARLA

Paramètre	Valeur
Mode	Synchrone à 20 FPS (fixed_delta_seconds = 0.05s)
Carte	Town01 (extensible à toutes les cartes CARLA)
Météo	Conditions claires (cloudiness=0, precipitation=0)
LIDAR	16 secteurs, portée 50m, 56000 points/sec
Obstacles	Maximum 3 obstacles par épisode

Annexe C : Références Bibliographiques

Algorithmes d'Apprentissage par Renforcement :

- Schulman et al., "Proximal Policy Optimization Algorithms", 2017

- Schulman et al., "High-Dimensional Continuous Control Using Generalized Advantage Estimation", 2016

Simulateur :

- Dosovitskiy et al., "CARLA: An Open Urban Driving Simulator", 2017

Conduite Autonome :

- Kiran et al., "Deep Reinforcement Learning for Autonomous Driving: A Survey", 2021

Frameworks :

- Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library", 2019
- Documentation CARLA : https://carla.readthedocs.io/