Rapport de stage de 3ème année de BUT Informatique

Optimisation de la trajectoire du laser en fabrication additive à l'aide de techniques d'apprentissage par renforcement profond

Du 24 février 2025 au 31 mai 2025

Tuteur: Kromer Robin

Référent: Marty Sidonie

Année Académique: 2024-2025

Par Martins Hugo

Résumé

Dans le cadre de mon stage à l'I2M de l'Université de Bordeaux, intégré au sein du département MPI, Materiaux-Procedes-Interactions, j'ai pu travailler sur l'optimisation du laser utilisé dans le processus LPBF[1] de fabrication additive. La fabrication additive, plus connue sous la catégorie d'impression 3D, permet la création de pièces par ajout de matières, couches après couches. Utilisée dans des secteurs comme la médecine ou l'aérospatial, la qualité des pièces est cruciale, hors certains défauts de fabrication et certaines malformations sont possibles.

Dans le but d'y répondre, nous chercherons dans ce papier une réponse aux problèmes de porosité et d'irrégularité de couches de matière par le biais de l'intelligence artificielle. À l'aide de techniques d'apprentissage par renforcement profond, nous simulerons le laser utilisé dans le processus de fabrication et nous l'optimiserons, dans le but final d'être en mesure de modifier la trajectoire et la vitesse du laser en temps réel, afin d'éviter les complications précédemment mentionnées. Dans ce rapport, je présente l'implémentation de l'algorithme DQN dans notre cas précis, en s'inspirant du jeu *Snake* pour représenter le laser. Nous obtiendrons ainsi une IA capable de réaliser des trajectoires simples en maximisant sa vitesse, tout en étant capable de l'adapter en fonction de la répartition de température dans la pièce.

Abstract

In the context of my internship at Bordeaux's University I2M laboratory, and working under the MPI, Materiaux-Procedes-Interactions's department, I was able to work on optimizing the laser used in the LPBF additive manufacturing process. Additive Manufacturing, more frequently known under the name of 3D printing, allows for the creation of objects by adding material layer by layer. Used in fields like medicine and aerospace, the quality of the pieces produced is crucial, yet manufacturing defects and irregularities of layers still happen.

In order to answer this issue, we will seek to find a solution to the porosity and layers irregularities problems through artificial intelligence. Using Deep Reinforcement Learning technics, we will simulate the laser used in the process of additive manufacturing and we will optimize it in order to, as a future final objective, be able to modify the trajectory and speed of the laser in real time, allowing to avoid the issues previously mentionned. In this report, I will present the implementation of the DQN algorithm in this specific case, taking inspiration of the Snake game to represent the laser. We will obtain an AI capable of doing simple trajectories, maximizing its speed as well as being able to adapt how fast it goes thanks to the thermic repartition caused by the laser.

[1] Laser Powder Bed Fusion

Remerciements

J'aimerai exprimer mes remerciements à Monsieur Kromer pour son suivi, son aide quant au sujet du stage sur la partie physique et son attention quant à nos difficultés. Je souhaite aussi remercier Monsieur Da rosa pour ses nombreux conseils et aides pour tous les problèmes techniques liés au code que j'ai pu avoir, et pour m'avoir permis d'apprendre beaucoup de concepts et éléments clés dans un domaine dans lequel je souhaite me professionaliser. J'aimerais remercier William Milan, camarade de classe mais aussi membre de l'équipe lors de ce stage, avec qui il était facile de travailler. Enfin, j'aimerais exprimer ma gratitude à Madame Marty pour ses conseils et leur pertinence quant au rapport et à la présentation.

Table of contents

1 Contexte du stage

1.1 Intégration au laboratoire de l'I2M

Situé à côté de l'esplanade des Arts et Métiers de Talence, l'I2M, soit Institut de Mécanique et d'Ingénierie de l'Université de Bordeaux, est proche de l'IUT Informatique. Le laboratoire a des tutelles et partenariats avec différentes entités. Dans les tutelles académiques, on y retrouve le CNRS, l'INRAE, l'ENSAM, ou encore Bordeaux INP. Concernant les partenaires industriels, on retrouve le groupe Airbus, le CEA, ou encore Onera. Le laboratoire développe des solutions et effectue des recherches dans les sciences mécaniques, parmis lesquels on pourrait simplement citer l'énergie, l'infrastructure, ou les transports.

L'I2M a été créé en 2011 par regroupement de trois Unités Mixtes de Recherche et de trois Équipes d'Accueil, et a plusieurs domaines de recherches: DUMAS, APY, TREFLE, IMC, MPI et GCE. J'ai réalisé mon stage au sein du département MPI (Matériaux, procédés et interactions), ma mission portant sur le thème de la fabrication additive. Souvent connu sous le nom d'impression 3D, la fabrication additive est un concept plus large regroupant des procédés et concepts variés dont fait partie l'impression 3D. Ses applications, nombreuses et variées, sont présentent dans beaucoup de secteurs industriels, mais on la retrouve en particulier dans le domaine médical et dans le secteur aérospatial.

1.2 Environnement de travail

J'ai réalisé le stage dans un open space de l'I2M, avec des horaires assez flexible tant que sept heures de travail journalières étaient bien effectuées, et en autonomie pour la majeure partie du temps. J'ai été sous la direction de Monsieur Kromer, maître de conférence à l'Université de Bordeaux et specialisé dans la fabrication additive. Durant le stage, j'ai aussi eu l'aide et les conseils de Monsieur Da rosa, ingénieur en Machine Learning.

En arrivant pour la première fois à l'I2M, William Milan et moi fûment accueillis par Monsieur Kromer, qui nous a expliqué, à la manière d'un maître d'ouvrage, le but du travail à mener et ce que l'on serait amener à produire au cours de ce stage, ainsi que le rythme du travail que nous suivrons et les rendus attendus. Monsieur Kromer nous a aussi présenté Monsieur Da rosa, ingénieur en Machine Learning, afin que l'on ai quelqu'un vers qui se tourner pour toutes questions concernant la partie technique, c'est à dire tout ce qui concerne le code et l'apprentissage par renforcement profond.

Dès le départ, nous avions été informés que le stage comporterait deux étapes principales. Dans un premier temps, moi et William devrions travailler sur la tâche individuelle qui nous a été assignée. Dans un second temps, nous chercherons à mettre notre travail en commun, afin d'avoir une application finale englobant nos deux projets et permettant une interaction entre eux.

1.3 Projet

Concernant les missions assignées, j'ai pour ma part eu celle de développement d'une IA, ayant pour but l'optimisation et la simulation de la trajectoire du laser dans le processus LPBF de fabrication additive. Ajuster le parcours réalisé par le laser lors de l'impression d'une pièce pourrait pallier aux soucis de porosité et d'irrégularité des couches, et c'est avec cette idée qu'est née cette mission. La tâche attribuée à William Milan portait sur l'aspect thermique du processus, ses travaux étant en lien direct avec la propagation de la chaleur.

Une fois que nos travaux seront suffisamment avancés, le but sera de les combiner afin d'avoir une IA plus réaliste, simulant au mieux le laser lors de la fabrication additive, prenant en compte les propriétés physiques qui y sont liées. Cette IA, une fois complètement fonctionnelle, permettrait au laboratoire de pouvoir faire des simulations du procédé LPBF et d'optimiser en temps réel l'impression. En effet, il est pour l'instant encore impossible d'adapter automatiquement l'impression quand elle est en cours. Ce procédé est jusqu'à présent préprogrammé; le laser avance à vitesse fixe, les trajectoires sont déterminées à l'avance en fonction des pièces et la propagation de la chaleur n'est pas prise en compte, ce qui peut donner lieu à des impressions imparfaites. Une des couches imprimées pourrait ainsi être poreuse et, à cause de cela, affecter la qualité de la pièce de manière négative et impacter toutes les autres couches.

Afin d'arriver à cet objectif, j'ai principalement réalisé des activités de développement informatique au cours du stage. Comme le travail fût effectué en autonomie, je n'ai par conséquent pas eu à travailler en équipe, tout du moins pour toute la première partie du stage, soit avant la combination de mon travail et de celui de William. J'ai cependant présenté mon avancée tous les jours par le biais d'un compte rendu écrit sur Discord, et j'ai aussi pu présenter mes résultats de manière directe toutes les deux semaines à l'oral, en présence de Monsieur Kromer et de Monsieur Da rosa. Afin de s'assurer que le projet se déroule dans de bonnes conditions et commence sans encombre, la première semaine était consacrée à améliorer nos connaissances et compétences en apprentissage par renforcement profond, tout en prenant connaissance du projet et en l'assimilant. Après cela, le reste du temps était consacré au développement de l'IA et au projet dans sa globalité.

Pour mieux comprendre le contexte dans lequel s'inscrit le projet, ainsi que les relations entre les différents acteurs, vous pouvez vous référer au schéma ci-dessous:

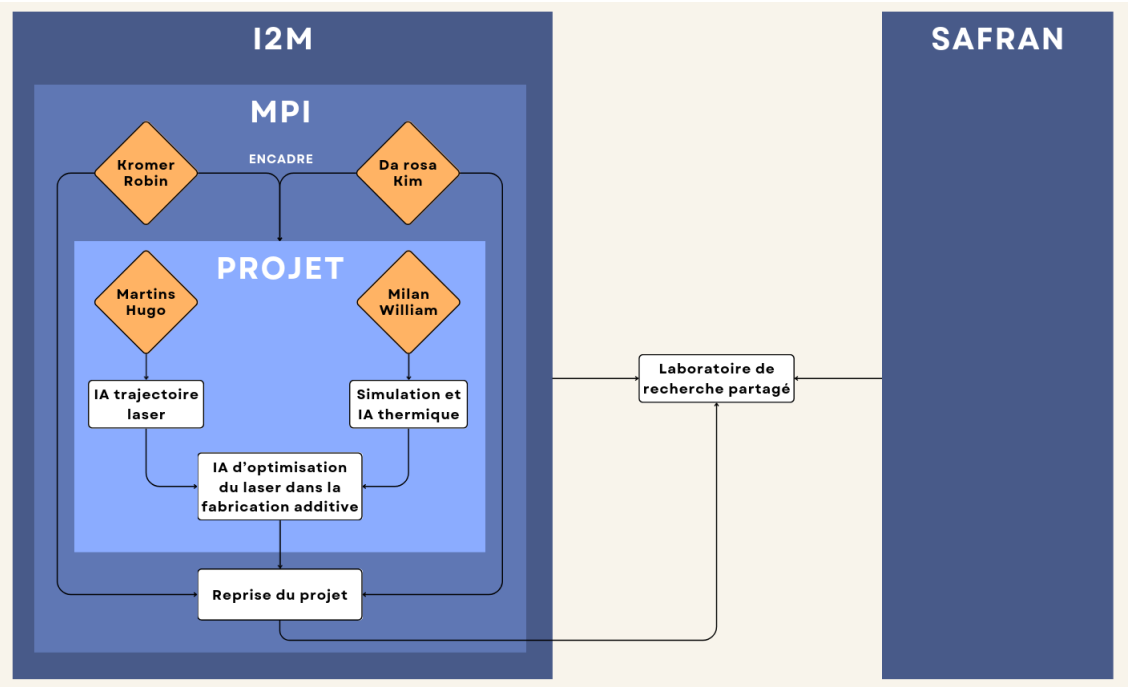

Figure 1: Schéma de l'environnement du projet

Au sein du projet, moi et William travaillerons en premier sur notre partie respective, avant de joindre nos codes pour finaliser le projet. Monsieur Kromer et Monsieur Da rosa nous encadreront, et récupéreront le code produit afin de le continuer, s'assurer de son fonctionnement et afin de l'améliorer, avant que celui-ci ne soit utilisé pour le laboratoire commun, partagé entre Safran et l'I2M.

2 Activité professionnelle

2.1 Environnement de l'IA et apprentissage par renforcement profond

2.1.1 Environnement Snake

Afin de simuler le laser dans le processus LPBF, par lequel il est possible d'imprimer des couches de matières, le projet fût basé sur le jeu *Snake*. Dans ce jeu, le joueur incarne un serpent, typiquement représénté par une ligne, et doit manger un maximum de fruits. On retrouve typiquement les aspects et règles suivantes :

- Le serpent apparaît à une certaine position sur l'écran, et est tourné vers une certaine direction. Aussi, un unique fruit est placé aléatoirement sur l'écran au départ.

- À chaque fois que le serpent mange un fruit, c'est à dire quand sa tête entre en contact avec, sa taille et le score augmentent. De plus, un nouveau fruit apparaît à une position aléatoire, de sorte à ce qu'il y en ait toujours précisément un dans le jeu.
- Si le serpent se heurte à une collision, qui peut être un mur ou son propre corps, la partie s'arrête.
- Le serpent va à vitesse constante, de case en case, et ses déplacements possibles sont limités aux nombre de 3, qui sont: aller à gauche, aller à droite, aller tout droit.

Dans notre cas, et dans un premier temps, le fonctionnement des fruits sera différent. Au lieu de n'avoir qu'une seule case contenant un fruit pour un moment T dans la partie, nous aurons dès le départ une multitude de ce type de case. De plus, aucun nouveau fruit n'apparaîtrat s'il y a un contact entre celui-ci et le serpent. Ce changement nous donne une condition de victoire, qui n'existait pas dans la version originale. En effet, si aucun fruit n'apparaît lorsque le serpent en mange un, alors il devient possible de tous les manger. Ce changement est dû au fait que l'on souhaite représenter la pièce à imprimer par les fruits. Par exemple, si l'on souhaite réaliser une forme carrée d'une certaine taille, disons 5 par 5 cases, un seul fruit ne suffira pas. On remplira donc un carré de 5 par 5 cases, qui représentera la forme à réaliser.

Aussi, nous ajouterons une nouvelle règle, qui est la terminaison automatique de la partie si le serpent n'a pas mangé de fruit après un certain nombre de tours. Celle-ci nous sera fortement utile afin d'éviter d'avoir des parties qui ne s'arrête jamais, dans le cas où le serpent tournerait en rond à l'infini.

2.1.2 Apprentissage par renforcement dans l'environnement Snake

L'apprentissage par renforcement est une technique d'apprentissage automatique, dans laquelle une IA interagit avec un environnement et s'adapte accordément. Si l'IA a connaissance de l'état dans lequel elle se situe, et si elle obtient des récompenses en fonction des actions qu'elle effectue, alors elle est capable d'apprendre. On distingue ainsi les concepts majeurs en apprentissage par renforcement, que sont les **récompenses**, les **actions**, et les **états**. Il faut aussi savoir que le but est de maximiser les récompenses dans le temps, afin d'avoir la plus haute récompense cumulée possible à la fin, et pas simplement la plus haute récompense à un instant T. Dans notre cas d'application, les récompenses que l'on peut obtenir à la suite d'une action sont assez intuitives :

1. Si le serpent mange un fruit, alors il obtient une récompense positive.
2. Si le serpent ne mange rien, ou plus précisément s'il se déplace sur une case vide, alors il ne gagne aucune récompense.
3. Dans le cas où, à la suite d'une action, le serpent se heurte à une collision, alors il aura une récompense négative. On parlera ici de pénalité.

Comme dit précédemment, afin de pouvoir apprendre, l'IA doit être capable de savoir dans quel état elle se situe. Intuitivement, on peut voir ça comme un snapshot, un instantané contenant diverses informations de l'environnement, qui lui serait utile dans sa prise de décision. Contrairement aux rewards, les informations contenues dans l'état ne sont pas forcément celles auxquelles on pourrait s'attendre. Dans un cas comme Snake, on pourrait s'attendre à ce que l'IA connaisse les informations du contenu de chaque case du jeu, mais cette idée ne sera pas choisie ici. À la place, nous aurons les informations suivantes:

- La présence d'un danger, soit d'une collision, dans les cases atteignables autour du serpent.
- La direction du serpent, c'est à dire vers où il se dirige.
- La ou les directions vers le fruit le plus proche. Il peut y avoir jusqu'à deux directions, dans les cas où le fruit est en diagonale par rapport au serpent. Si on représente ça par une matrice, on peut le voir comme la ligne et la colonne contenant le fruit étants toutes deux différentes du serpent.

Il reste encore quelques concepts qui n'ont pas été mentionnés, comme le choix des actions et les phases d'exploration. Ceux-ci seront expliqués progressivement si nécessaire, cette section étant consacrée aux aspects les plus essentiels en apprentissage par renforcement.

2.1.3 Agent Deep Q-learning

Le Deep Q-learning est un algorithme d'apprentissage par renforcement profond. On peut le voir comme une amélioration directe du Q-learning, combinant cet algorithme d'apprentissage par renforcement avec les réseaux de neuronnes, octroyant ainsi la dimension de deep learning. Nous avons précédemment présenté comment l'apprentissage par renforcement serait appliqué dans notre cas, c'est pourquoi nous expliquerons brièvement ici le principe de cet algorithme, en commençant par expliciter sa différence avec le Q-learning:

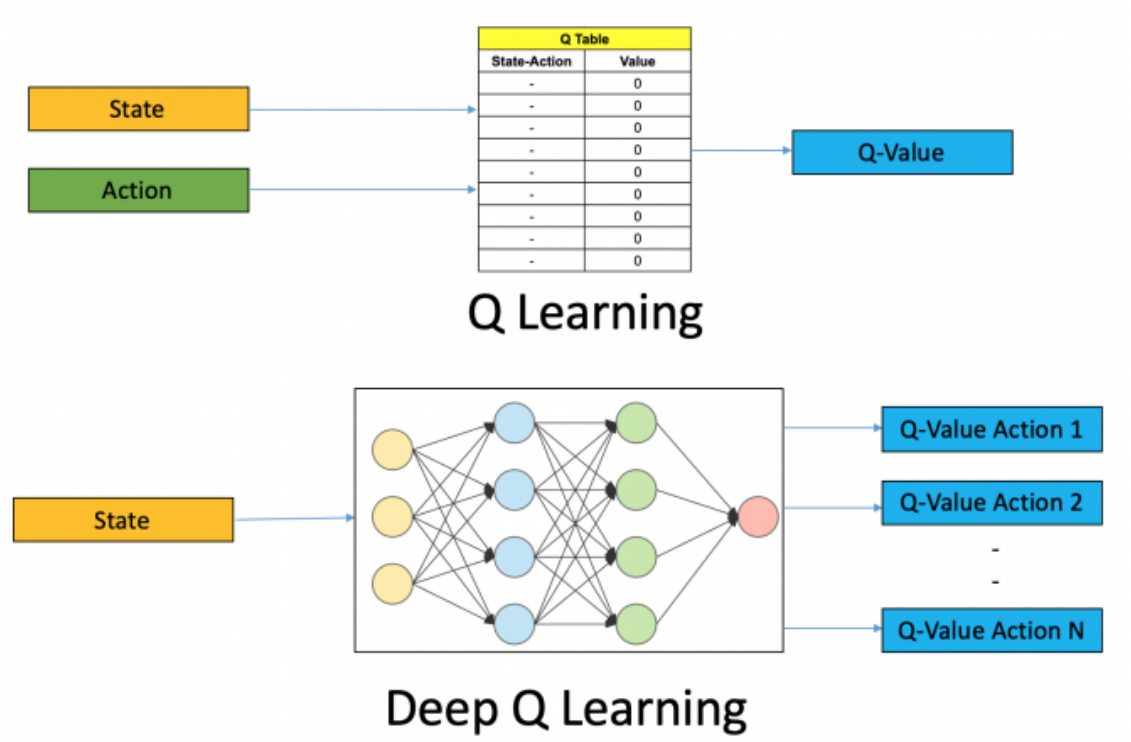

Figure 2: Comparaison du Q-learning et du Deep Q-learning[2]

Le premier cas illustre le fonctionnement du Q-learning. Dans cet algorithme, nous disposons d'une Q-table, répertoriant les valeurs de chacunes des paires état-action. Initialement, une valeur de base est prédéfinie pour chaque paire dans le tableau. Concrètement, l'action sera choisie de la manière suivante:

1. On récupère l'information de l'état dans lequel l'IA se trouve.
2. Hors phase exploratoire, soit en phase d'exploitation, on parcours toutes les paires état-action possible depuis cet état et retourne l'action qui donne la meilleur Q-value. Autrement dit, à partir de cet état, on va regarder pour chaque action possible laquelle donne le meilleur résultats, et c'est celle-là que l'on choisira
3. En phase exploratoire, on choisi une action aléatoire parmi celles possibles.

Dans le cas du Deep Q-learning, cette Q-table est remplacée par un réseau de neuronnes. En entrée du réseau, nous pouvons retrouver l'état; à la sortie, nous n'avons plus une seule et unique valeur retournée mais à la place N valeurs différentes, qui correspondent à la récompense estimée pour chaque action. En somme, le réseau de neuronnes récupère les données de l'état, les interprètent, puis retourne les estimations de Q-value pour chaque action. Il ne reste ainsi plus qu'à choisir l'action correspondant à la Q-value estimée la plus haute. Similairement au Q-learning, cela n'est vrai que pour la phase d'exploitation. Si on est en phase d'exploration, on choisi simplement là encore une action aléatoire parmi celles possibles.

[2]source: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

Les avantages de cette évolution sont majeurs. En remplaçant la table par un réseau de neuronnes, l'algorithme est capable d'éviter les problèmes de dimensionnalités que possèdent son prédécesseur, et il devient aussi possible d'avoir des valeurs flottantes dans l'état, qui était originellement contraint à des nombres entiers.

Cela peut être illustrer avec l'exemple qui suit. Présumons que nous n'ayons rien changé aux actions, mais que dans notre état nous connaissons désormais le contenu de chaque case de la grille du jeu. Pour une grille de 100 par 100 cases, cela nous donne un total de 10000 cases. De plus, chaque case peut avoir 3 valeurs différentes, qui sont vide, fruit, ou collision. On remarque ainsi les choses suivantes:

- Avec Q-learning, cela donnerait lieu à $3^{10000} * 3$ lignes pour notre Q-table. En effet, nous avons 10000 cases et chacune peut avoir 3 valeurs différentes, ce pourquoi on retrouve 3^{10000}. Enfin, on multiplie par 3 car c'est le nombre d'actions possibles.

- Avec Deep Q-learning, le calcul est différent: présumons ici que le réseau de neuronnes est une couche d'entrée correspondant au nombre de cases, une couche intermédiaire de 250 neuronnes et une couche de sortie correspondant au nombre de cases. Cela nous donne le calcul $10000 * 250 + 250 * 3$, ce qui est égal à 2 500 750.

Il devient simple de voir l'importance du réseau de neurones. Avec Q-learning, le nombre de lignes de la Q-table est tellement grand que ma calculatrice n'arrive pas à le calculer. Avec le réseau de neuronnes, cela est réduit à *seulement* environ 2 500 000 paramètres.

2.2 Environnement réaliste et optimisation de l'IA

2.2.1 Amélioration de l'environnement

Afin de représenter au mieux le laser utilisé dans les processus de fabrication additive, il a été décidé d'améliorer l'environnement et de modifier les règles afin qu'elles soient plus proches de la réalité. Le laser, représenté par le serpent, pourra désormais accélérer et ralentir, ainsi que tourner selon des angles de 45° et 135°. Naturellement, il garde aussi les actions mentionées au départ, qui sont maintenir sa vitesse, aller tout droit, et tourner à 90°.

Pour qu'il soit plus simple de comprendre comment ces changements vont affecter l'IA, les règles du jeu sont listées ci-dessous:

- Le laser apparaît sur un bord de la pièce à faire, le serpent est donc directement à côté des fruits. On peut aussi le voir comme étant déjà placé sur la pièce. Par conséquent, ni le placement du serpent ni celui des fruits n'est aléatoire.

- Le laser peut se déplacer en diagonale et aller à 3 vitesses différentes, allant de 1 à 3. Par exemple, en allant à vitesse 2, cela signifie que le serpent parcourt 2 cases en une seule action.

- Il a aussi fallut inclure une notion de drift, afin d'avoir des comporements suffisamment réalistes. Par exemple, si on avance à vitesse 3 et qu'on tourne à un angle de 90° ou 135°, il est simple d'imaginer que cela soit irréaliste et que le laser dévie. Pour prendre un virage en épingle dans des routes de montagne, il faut ralentir, car si on roule vite et qu'on tourne brusquement, on dérape. Ici le principe est le même, donc j'ai décidé d'établir des règles et conditions faisant en sorte de représenter ce phénomène. Sans entrer dans les détails, elles peuvent être comprises comme suit: plus on va vite, et plus on tourne brusquement, plus le laser déviera de la trajectoire choisie initialement.

Maintenant que ces nouvelles règles ont été expliquées, il est simple de présenter plus concrètement leur impact sur les actions et sur le contenu de l'état de l'IA.

- Il y a 10 actions possibles; 7 concernent les directions, et 3 concernent la vitesse. À chaque tour, l'IA choisira une action parmi la vitesse, et une action parmi la direction.

- L'ajout de ces actions a entraîné des changements dans l'état. En effet, un changement des règles du jeu influencera l'environnement, ce qui à son tour peut influencer ce qu'à besoin de connaître l'IA, soit

le contenu de l'état. Initialement, le serpent était limité à la vitesse 1, et ne pouvait aller que dans 3 directions différentes. Les cases atteignables par le serpent étaient donc au nombre de 3, et on avait l'information du contenu de ces cases dans l'état. On comprend donc que depuis l'ajout de la vitesse, du drift et des directions diagonales, il a fallu rajouter dans l'état le contenu de toutes les cases dans un rayon de 270° et de 3 cases autour du serpent, car celles-ci sont devenues atteignables. De manière encore plus évidente, le serpent n'avait pas différentes vitesses au départ. Comme on en a ajouté, il a fallu mettre dans l'état l'information sur la vitesse du serpent.

Dans ce nouvel environnement et avec ces nouvelles informations, le serpent pourra plus précisément imiter le processus LPBF. Dans ce nouveau terrain de jeu, nous chercherons à optimiser l'IA. Plus précisément, le but sera désormais d'avoir un serpent capable de manger tous les fruits le plus rapidement et précisément possible. Par exemple, si l'on souhaite réalisé une forme comme un carré remplie, le serpent devra parcourir toutes les cases du carré sans sortir à l'extérieur et le plus vite possible. On verra qu'afin d'y arriver, il faudra établir de nouvelles récompenses. En effet, si la seule récompense est celle de manger un fruit, le serpent n'aura pas d'incitations à accélérer par exemple.

2.2.2 Nouvelles récompenses

Afin de pouvoir accélérer, une première idée serait de donner des récompenses au serpent selon sa vitesse. Donner des récompenses plus hautes quand le serpent est rapide permettra de lui faire comprendre qu'aller vite est une bonne chose. Cependant si nous implémentions directement cette idée comme tel, on risque de rencontrer un problème majeur.

1. Rappelons que le but de l'IA est d'obtenir la récompense la plus haute possible dans le temps.
2. En tenant compte de cela, on notera que les récompenses associées aux fruits sont finies, car il en existe un nombre limité. Cependant, on observe que celles associées à la vitesse ne le sont pas. Autrement dit, le serpent pourrait gagner des récompenses toute la partie par le simple fait d'aller vite. Sans le concept d'interruption de partie mentionné auparavant, s'activant lorque le serpent n'a pas mangé de fruits depuis un certain nombre de tours, cette récompense serait même infinie.

De ce fait, nous pouvons déduire que cette idée à des désavantages majeurs. La récompense de vitesse est simple à activer, car il suffit d'aller vite, et elle n'est pas finie, contrairement à celle des fruits. En conséquence, cette implémentation de la récompense de vitesse risquerait de produire une IA qui ignore complètement les fruits et se contente d'aller vite, tournant en rond jusqu'à ce que la partie se finisse. Ceci étant l'opposé total de ce que nous cherchons, j'ai décidé de lier la récompense de vitesse avec celle des fruits. Grâce à cela, le serpent gardera en information principale que les fruits sont la source de récompense, et il découvrira qu'aller vite en les mangeant permet d'obtenir de meilleurs résultats.

Le serpent étant désormais capable d'aller vite et de se diriger vers les fruits, on pourrait s'attendre à ce qu'il n'y ait plus de problèmes à régler. Ce n'est cependant pas le cas, car bien que l'IA ait appris à aller vite, elle n'a cependant pas appris à ralentir, ce qui est un problème majeur dans le cas où elle devrait effectuer des virages supérieurs ou égaux à 90°. En effet, comme mentionné précédemment, une fonction de drift qui cause une déviation de la trajectoire est appliquée au serpent s'il va vite et s'il tourne brusquement. Il est essentiel que le serpent apprenne à ralentir dans ces cas, ce pourquoi nous avons pensé au concept d'une vitesse de sécurité:

1. Présumons que le serpent vienne de manger un fruit, c'est à dire qu'il se situe sur une ancienne case fruit au début du tour, avant d'effectuer son déplacement. On peut comprendre ça comme le laser se situant au sein de la pièce à imprimer. Un aspect intéressant à noter est que ce cas est vrai à l'initalisation de la partie.
2. Dans un tel cas, on calculera une distance, le nombre de cases que le serpent peut parcourir, directement en face de lui, jusqu'à ce qu'il atteigne une collision.

3. Á partir de cette distance, et en connaissant le nombre de cases que le serpent parcours avant d'être à vitesse basse, que l'on peut voir comme la distance de freinage, il est possible de calculer la vitesse de sécurité. Si on respecte celle-ci, nous obtenons une récompense positive. Dans le cas inverse, si nous sommes en excès de vitesse, on reçoit une pénalité.

Cette récompense est facilement explicable en prenant l'exemple d'une voiture. Si une voiture roule à 90km/h alors qu'il y a un mur à 15 mètres en face d'elle, il est physiquement impossible qu'elle ait le temps de freiner et de l'éviter. Cependant si elle allait par exemple à 30km/h, cela serait possible. C'est ce principe qu'on applique à notre serpent; il reçevra une pénalité dans le premier cas, et une récompense dans le second, à partir du moment où il se situe sur la route, soit dans la figure à imprimer.

2.2.3 Benchmark et sauvegardes des résultats

Nous avons désormais tous les éléments à notre disposition. Notre serpent est capable d'aller à des vitesses variées, de tourner à différents angles, et il peut comprendre qu'il faut maximiser la vitesse tout en étant raisonnable, afin de ne pas dépasser la vitesse de sécurité. L'IA sera donc en capacité de comprendre, mais nous ne savons pas encore quelle importance il faut donner à chaque élément. Est-ce que toutes les récompenses et pénalités devraient être égales ? Devrait-on insister sur la pénalité de collision, afin de s'assurer que le serpent évite les murs, ou bien faudrait-il au contraire insister sur la récompense de vitesse afin de s'assurer qu'il maximise bien cette dernière et qu'il ne soit pas trop restreint par la vitesse de sécurité ?

Dans le cas présent, nous pouvons avoir des intuitions sur quels récompenses, quels paramètres il faudrait privilégier ou non, mais nous n'avons pas les moyens d'argumenter leur véracité, et encore moins les moyens de les prouver. C'est pourquoi, afin de pouvoir vérifier ce que l'on pourrait croire, et dans le but de facilement comprendre et expliquer le comportement de l'IA, il a été décidé de mettre en place un système de benchmark. Le but est de tester différentes combinaisons de paramètres sur différents cas de parcours de laser à effectuer. Ce que j'appelle combinaison de paramètre pourrait être représenté par le groupe suivant :

```
Récompense de fruit = 3.5, Pénalité de collision = 10.0, Récompense de vitesse = 1.5,
                    Récompense/Pénalité de vitesse de sécurité = 1.0
```

Ceci est une combinaison de paramètres parmis tant d'autres. On associe certaines valeurs à certaines rewards, et j'appelle ce regroupement une combinaison de paramètres (de récompenses). On notera que, naturellement, chaque combinaison est unique, et que chaque paire Récompense-Valeur est elle aussi unique au sein d'un même benchmark, le but étant de tester différentes combinaisons différentes.

Le moyen choisi pour générer un champ de paramètre quasi-aléatoire est LHS[3]. C'est une méthode de random sampling qui nous permettra d'avoir des combinaisons de paramètres différentes, choisies de manière quasi-aléatoire afin d'éviter des problèmes de l'aléatoire classique. Par exemple, si l'on choisit complètement aléatoirement 10 nombres entre 0 et 5, il se peut qu'ils aient tous une valeur proche de 5, même si c'est peu probable. La méthode LHS permet de diviser cet espace en 10 intervalles, et de prendre une valeur de chaque intervalle au hasard, ce qui permet d'avoir une répartition sans trous importants tout en gardant de l'aléatoire.

Pour comprendre le contenu du benchmark, vous pouvez vous réferer à l'image ci-contre. Il y aura 9 cas de test différents, correspondants à 3 distances différentes et 3 trajectoires différentes. Les trajectoires seront : linéaire, angle de 135°, et angle de 90°. Les distances quant à elles seront de 10, 30, et 90 cases. Le but sera d'analyser l'influence de chaque paramètres pour chaque parcours, de comprendre quel aspect du benchmark est le plus important, et d'au final être capable de, dans une certaine mesure, déterminer les valeurs à affecter à chaque paramètre selon le type de parcours à réaliser.

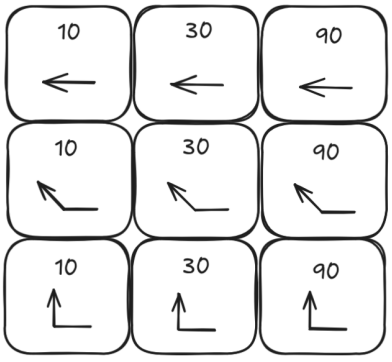

Figure 3: Parcours du benchmark

[3]Latin Hypercube Sampling

Afin de pouvoir recueillir les informations obtenues au cours de l'exécution du benchmark, j'ai décidé de mettre en place un système de logging, afin d'obtenir des informations variées sur l'IA au fil des parties. Cela peut être séparé en deux points:

- D'une part, j'enregistre dans un fichier de log journalier les résultats, scores et autres métriques de l'IA obtenues au fil des parties, ainsi qu'à quelle configuration et quel parcours du benchmark ces résultats correspondent. Ainsi, si je décide de tester 10 combinaisons de paramètres différents, que l'on peut comprender comme 10 réglages de l'IA différents, ce fichier me permettra d'enregistrer les performances de chaque IA.
- D'autre part, j'enregistre pour chacuns de ces parcours à réaliser, soit pour les 9 parcours du benchmark, la meilleure partie réalisée. Dans ces fichiers, il sera donc possible de voir quels paramètres ont permis d'obtenir les meilleurs performances. Ainsi, il suffira de les consulter pour immédiatement connaître la meilleure combinaison de paramètre trouvée parmi toutes les combinaisons de paramètres testées lors du benchmark.

Afin de comprendre la partie sur l'analyse de données, il peut être fortement utile de voir le deuxième point, soit les 9 fichiers de logs contenant les données des meilleurs IA par parcours, comme étant une ligne précise du premier fichier de log général comportant les informations de toutes les IA et de tous les parcours au fil du temps.

2.2.4 Analyse de données et recherche des meilleurs paramètres

Comme nous avons pu le voir, nous sommes désormais en capacité de tester des combinaisons de paramètres variées sur un benchmark de différents parcours, afin de pouvoir optimiser les récompenses de l'IA. Il est possible de paramétrer l'IA directement à partir des meilleurs combinaisons de paramètres trouvées pour chaque parcours, enregistrées dans les fichiers de logs mentionné lors du dernier point. Cependant, il serait préférable de chercher à savoir quels sont vraiment les paramètres qui donnent les meilleurs résultats.

Il faut en effet comprendre qu'avec ces données, nous ne connaissons que les meilleures combinaisons de paramètres par parcours trouvées. Nous ne testons pas toutes les combinaisons possibles, donc on ne peut pas prétendre que celles trouvées sont les meilleurs atteignables. Dans le but de déterminer celles-ci, nous devrons procéder à l'analyse de données du fichier de log du benchmark entier, celui contenant les résultats et l'évolution de chaque IA, pour chaque parcours. Afin de bien comprendre son organisation, observons en guise d'exemple la ligne suivante:

```
IA numéro 1; paramètres food 1.0 - collision 10.0 [...]  ; parcours 1; game 200; avg_score
                                    27.93; avg_speed [...]
```

On peut la séparer en plusieurs points: les informations sur l'IA, les informations sur où en est la partie et quel parcours on effectue, et enfin quels sont les scores, statisitiques, métriques qui y sont associées. Ce fichier de log, faisant dans mon cas près de mille lignes et comportant une dizaine de métriques de score différentes, est naturellement inutilisable en tant que tel. En conséquence, il a fallu que j'analyse les données du fichier, et j'ai ainsi :

1. Transformer les données du fichier de log en un dataframe avec Pandas, une librairie Python.
2. Traiter les données, avec principalement: la gestion des valeurs manquantes, la gestion des valeurs aberrantes s'il y en a, et la normalisation des données si nécessaire.
3. Analyser les données: matrices de corrélations et clustermaps, graphiques en radar, techniques de réduction de dimensionnalité comme T-SNE[4] et PCA[5], random forest regressor et optimisation des paramètres.

Je n'entrerai pas dans les explications pour les deux premiers points, étant donné qu'on peut le résumer simplement par s'assurer que les données soient proprement gérées pour pouvoir proprement les analyser. En effet, si les données sont mauvaises et pas traitées, les graphiques seront au mieux impossibles à interpréter et au pire faux et mal interprétés. Une fois cette phase réalisée, j'ai donc pu analyser ces données.

[4]t-distributed Stochastic Neighbor Embedding
[5]Principal Component Analysis

Concernant les matrices de corrélations et les clustermaps, je les ai utilisé dans le même but, à savoir déterminer les informations inutiles dans le dataframe, ainsi que les corrélations avec le score et la vitesse. Présumons que nous ayons deux métriques différentes, qui seraient le score moyen et le nombre de parties gagnées. Si l'on observe une corrélation haute, soit proche de 1, entre ces deux métriques, cela signifie qu'elles sont liées; lorsqu'une métrique augmente ou diminue, l'autre fait de même. Si jamais la corrélation était fortement négative, soit aux alentours de -1, cela signifierai qu'elles sont anti-corrélées. Autrement dit, lorsqu'une monte, l'autre descend. Dans le cas où la corrélation serait faible, vers 0, on comprend donc que cela indiquerait alors qu'il n'y a pas de relations entre les 2.

J'ai utilisé ces méthodes afin de pouvoir faire deux choses. D'une part, je m'en suis servi pour éliminer les métriques trop similaires. Par exemple, celles mentionnées précédemment étaient fortement liées et avaient des interactions similaires quel que soit le parcours. J'ai donc pu supprimer la métrique de nombre de parties gagnées car elle n'avait pas d'intérêt et rendait le dataset plus compliqué à lire. Le clustermap était notemment utile pour voir ces clusters, regroupements de corrélations, qui montraient rapidement quelles features du dataset étaient corrélées. D'autre part, j'ai utilisé les matrices de corrélations afin de voir quels paramètres de l'IA avaient une influence sur le score et la vitesse. Ici, on retrouve certaines de ces corrélations sous la forme d'un graphique en radar plutôt que sous la forme de matrice :

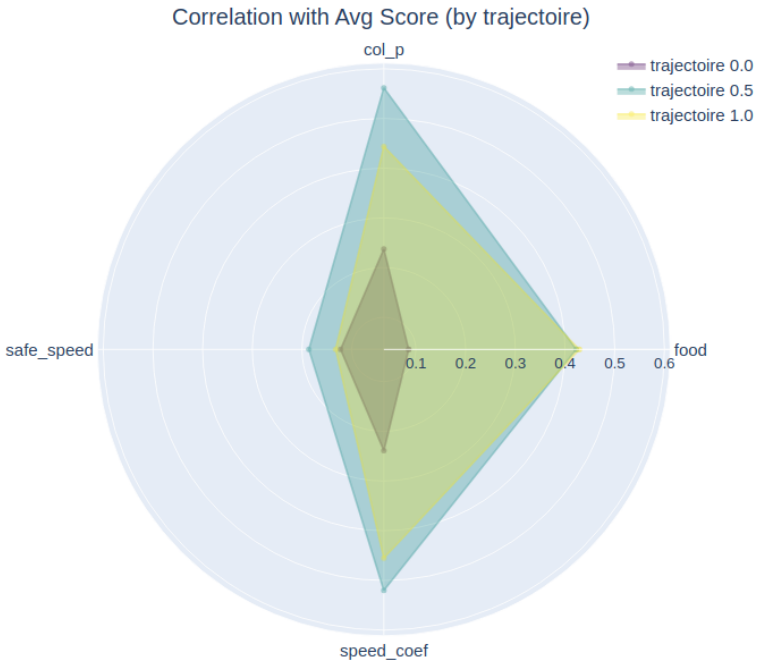

Figure 4: Radar Plot des corrélations des paramètres avec le score moyen

Sur ce graphique, on peut observer les corrélations entre les 4 paramètres du modèle, concernant la pénalité de collision, la récompense de fruit, la récompense de vitesse et la récompense/pénalité du respect de la vitesse de sécuité. J'ai effectuer un trie par trajectoire, et on note que celles-ci ont été normalisées, initialement ayant les valeurs 0, 1 et 2 respectivement. On remarquera surtout que les trajectoires 1 et 2 seraient similaire, tout du moins du point de vue de l'influence des paramètres sur le modèle, et que la trajectoire 0 serait isolé et sans rapport aux autres. On retrouve en effet ça dans l'exécution du benchmark: l'IA n'avait aucun mal à réussir cette trajectoire, atteignant rapidement le score maximal quelle que soit la distance. On comprend bien que les paramètres n'ont pas de réelles influences sur cette trajectoire car elle est trop simple, au contraire des deux autres.

J'ai ensuite utilisé les techniques de réduction de dimensionnalité afin de rapidement voir quels groupements ressortaient le plus, et voir si on pouvait là encore rassembler les parcours 1 et 2. Expliquons d'abord plus en détail en quoi cela consiste et ce que j'entends par groupement. Nous avons un jeu de données avec beaucoup de features différentes, dans mon cas il y en a plus de vingt. Il est impossible de chercher à comprendre les relations entre chaque données du dataset dans un espace à plus de 20 dimensions, c'est pourquoi ces méthodes peuvent êtres utilisées afin de déterminer les relations, linéaires ou non, tout en réduisant l'espace à 2 ou 3 dimensions afin que cela soit interprétable.

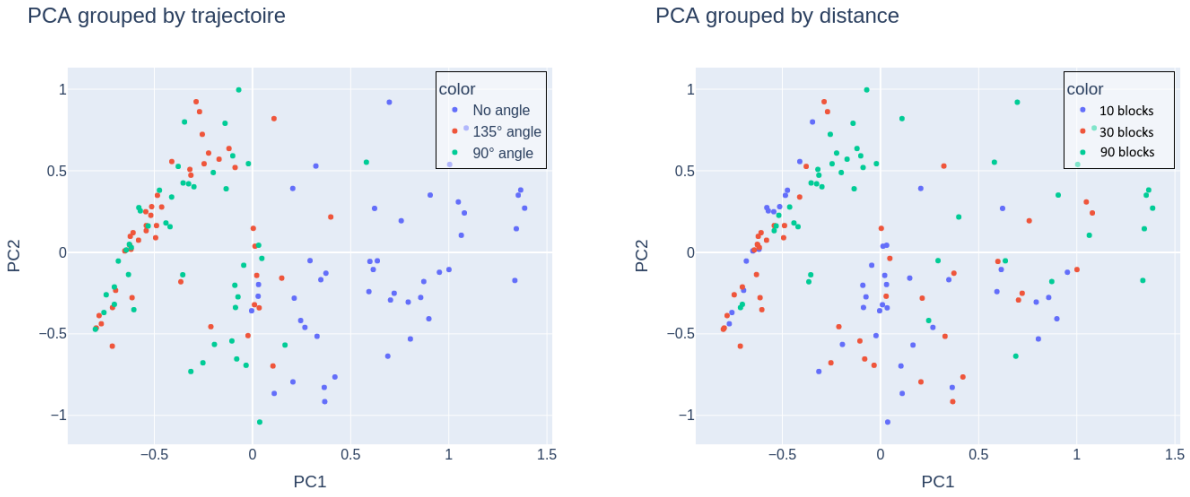

Figure 5: PCA avec groupement selon la distance et la trajectoire

Ci-dessus l'application de PCA, où chaque point représente une donnée du dataset. De plus, il est colorié selon le parcours ou la distance, dans le but de voir si des groupements ressortent. Sur le graphique à gauche, on remarque qu'on peut regrouper les parcours en 2 catégories: ceux avec un angle et ceux sans angles; les points verts et oranges sont mélangés mais pas ceux bleus, qui sont plus regroupés à droite. Sur le graphique à droite, il est impossible de déterminer quelconques groupements; peu importe la distance, les points sont tous plus ou moins mélangés. Avec ces observations, on comprend que les trajectoires ont le plus d'impact, donc c'est sur cet aspect qu'il faudra qu'on se concentre. On cherchera dans la prochaine étape à maximiser le score selon la trajectoire, étant donné que la distance est moins importante. En guise de note, on précisera tout de même que PCA permet de voir les relations linéaires, et que t-SNE permet de voir celles non linéaires. Par conséquent, on aurait sûrement vu des résultats et groupements différents si on avait utilisé t-SNE plutôt que PCA, indiquant des relations non-linéaires concernant la distance.

Une Random Forest est un modèle de Machine Learning, dont le principe est de combiner une mutltitudes d'arbre de décision, chacun prédictant un résultat. Si c'est un modèle de classification, la prédiction finale est choisie à la suite d'un vote à la majorité, où la classe prédite le plus est celle choisie. Dans notre cas, nous chercherons à prédire le score moyen obtenue par l'IA à partir de ses paramètres. Nous avons donc ici une valeur numérique flottante, qui ne sera par conséquent pas interprétable par un modèle de classification. Nous utiliserons alors un modèle de régression, où le résultat final prédit sera la moyenne des résultats prédits par chaque arbre du modèle de forêt aléatoire.

Suivant les déductions faites précédemment, j'ai séparé le dataset en trois parties, chacune correspondant à une trajectoire différente. Je n'ai conservé que les features des paramètres de l'IA ainsi que la feature concernant le score moyen, et j'ai ensuite créé un modèle de Random Regression Forest pour chacun des trois datasets. Le but est le suivant: pour chaque trajectoire, on utilise une forêt aléatoire pour prédire le score moyen à partir des paramètres.

Présumons que nos trois modèles soient très performants et efficaces, autrement dit qu'ils parviennent

à prédire avec précision les scores moyens de l'IA à partir de ses paramètres. Dans un tel cas, on pourrait essayer d'optimiser les paramètres du modèle DQN de Deep Reinforcement Learning directement à partir du modèle de la Random Forest. La manière de le faire pourrait être exactement la même que celle avec laquelle on fait le benchmark, simplement à une très grande échelle. Dit plus simplement:

- On recherche les meilleurs paramètres avec le modèle DQN; comme le processus est long, on ne lance qu'une vingtaine à une trentaine de combinaisons différentes.
- On obtient ainsi le fichier de log, que l'on transforme en dataset qu'on traite et analyse. Nous arrivons donc au point où nous avons une forêt aléatoire très performante, capable de prédire le score que l'on obtiendra simplement à l'aide des paramètres de l'IA DQN (qui sont donc les features du modèle de forêt aléatoire).
- On peut à ce stade effectuer une nouvelle recherche de paramètre, mais sur la forêt aléatoire et non sur le modèle DQN. La raison de procédé ainsi et que cela nous permettrai un immense gain de performance et de temps. Il est en effet extrêmement simple et rapide de tester 10000 combinaisons de paramètres différentes pour la forêt aléatoire, cela prendrait moins d'une minute. Cela serait absolument impossible à faire avec le modèle DQN, qui met plus d'un jour à faire seulement 30 combinaisons différentes.

On se serivirait ainsi du modèle de fôrêt aléatoire pour optimiser les paramètres du modèle DQN, utilisé donc comme le Surrogate Model du DQN. Malheuresement, j'ai du abandonné cette idée car les résultats de la forêt aléatoire n'était pas suffisamment satisfaisants pour qu'elle soit exploitable et représentative du DQN.

2.3 Données thermiques et nouvel objectif

2.3.1 Prise en compte de l'aspect thermique

Le procédé LPBF consiste en la création de pièces à via la fusion de poudres métalliques par laser. Celui-ci fusionne les poudres en appliquant de fortes températures en points précis, permettant ainsi, couche après couche, de créer la pièce souhaitée. De ce fait, un aspect qui n'a pas été pris en compte jusqu'à présent concerne la partie thermique du procédé, point extrêmement important étant donné que la répartition de chaleur est l'élément principal influençant la qualité d'une pièce. Notre but sera donc désormais de prendre cet aspect en compte, afin de pouvoir développer une IA qui est capable d'optimiser la trajectoire du laser afin d'être précis, rapide, et de fournir une répartition de chaleur équitable pour éviter les problèmes de malformations des pièces et les problèmes de porosité.

C'est pour cet objectif final du projet qu'intervient la partie développée par William. En effet, son travail porté en partie sur la simulation du bain de fusion, qui est créée par le laser dans le procédé LPBF. Le bain de fusion représente la fonte du métal, mais on interprètera ça comme la propagation de chaleur causée par le laser. En transmettant la trajectoire parcourue par le serpent à la fonction de la simulation du meltpool, William sera capable de me fournir la température, ou plutôt la valeur de bain de fusion en chaque case du parcour effectué.

Nous aurons donc de nouvelles informations à intégrer, ce qui se traduira par une nouvelle récompense qui prend en compte la température, et un nouvel état dans lequel on intégre ces nouvelles informations, dans une certaine mesure. Dans un premier temps, j'ai attribué à chaque case une température. Celles-ci seraient mise à jour lors de la récupération des informations du meltpool. Il n'existera dans notre cas que 3 types de températures, froid, tiède, et chaud, tout comme il n'existe que 3 vitesses différentes.

Avec ce nouvel environnement, nous avons donc une grille de cases comportant 2 valeurs, qui sont la présence de nourriture ou non ainsi que la température. Si le laser est lent et qu'il met du temps à faire le parcours, alors il va beaucoup chauffer la pièce, ce qui rendra lieu à des cases trop chaudes. Dans le cas contraire, on observe qu'un laser très rapide chauffe trop peu la pièce et donc nous n'avons presque que des cases froides. Avec cet évolution de l'environnement, la récompense à choisir afin de chauffer équitablement la pièce et n'avoir que des cases tièdes est assez intuitive.

- Si le serpent va vite dans des cases qui ont été jugées chaudes, c'est à dire s'il ajuste sa vitesse et cherche à éviter de chauffer trop longtemps les zones chaudes, alors il obtiendra une récomopense.

- Dans le cas inverse, s'il est lent dans les zones chaudes, leschauffant donc pendant longtemps, alors on lui donnera des récompenses négatives. Le même principe a été appliqué pour les cases froides. Les cases tièdes quant à elles n'ont pas de constraintes quelconques.

Il ne reste désormais plus qu'à intégrer les informations des températures dans l'état. J'ai choisi d'ajouter une valeur correspondant à la température moyenne des 3 cases autour du serpent, similairement à ce que j'avais fais pour la détection de collision, là encore dans un rayon de 270° en face du serpent car il peut tourner jusqu'à 135°, à droite et à gauche.

2.3.2 Boucle d'apprentissage

Nous avons donc les informations sur la propagation de chaleur que le parcours à causé, et nous avons modifié le code en conséquence afin que l'IA puisse apprendre de ce changement. Les pièces sont réunies et il ne reste plus qu'à les assembler: nous allons ici présenter l'évolution finale de l'IA, par la création d'une boucle d'apprentissage, qui sera l'intégration complète de mon travail avec celui de William, optimisant du début à la fin le laser dans le processus LPBF en prenant en compte un maximum les aspects physique de ce procédé.

Je vais désormais présenter ce programme final, qui peut être séparé en 3 parties:

1. Au lancement du code, mon IA s'entraîne une première fois sans prendre en compte l'aspect thermique avec les meilleurs paramètres qu'on a déterminé lors de la phase de benchmarking et d'analyse de données mentionné auparavant.

2. Dans un second temps, nous entrons dans la boucle d'apprentissage. Ici, l'IA va faire appel au code de William afin de déterminer les températures de chaque case que l'on a parcouru lors la meilleure partie. Si la chaleur n'est pas équitablement (ou presque) répartie, alors on récupèrera les données de la température des cases, que l'on intégrera dans un prochain entraînement de l'IA. Avec ces nouvelles informations, et avec la récompense de température, le serpent sera capable d'adapter sa vitesse accordément.

3. Enfin, lorsque la chaleur sera répartie de manière suffisamment équitable, ou après un certain temps écoulé, dans le cas où nous n'arrivons pas à atteindre une répartition de chaleur optimale, on utilisera l'IA de réglage de puissance de William. On peut voir cet IA final comme le dernier ajustement permettant de gérer la puissance du laser, et donc gérer la température pour produire des pièces sans défauts

Ces explications peuvent être plus facilement comprises par le schéma qui suit :

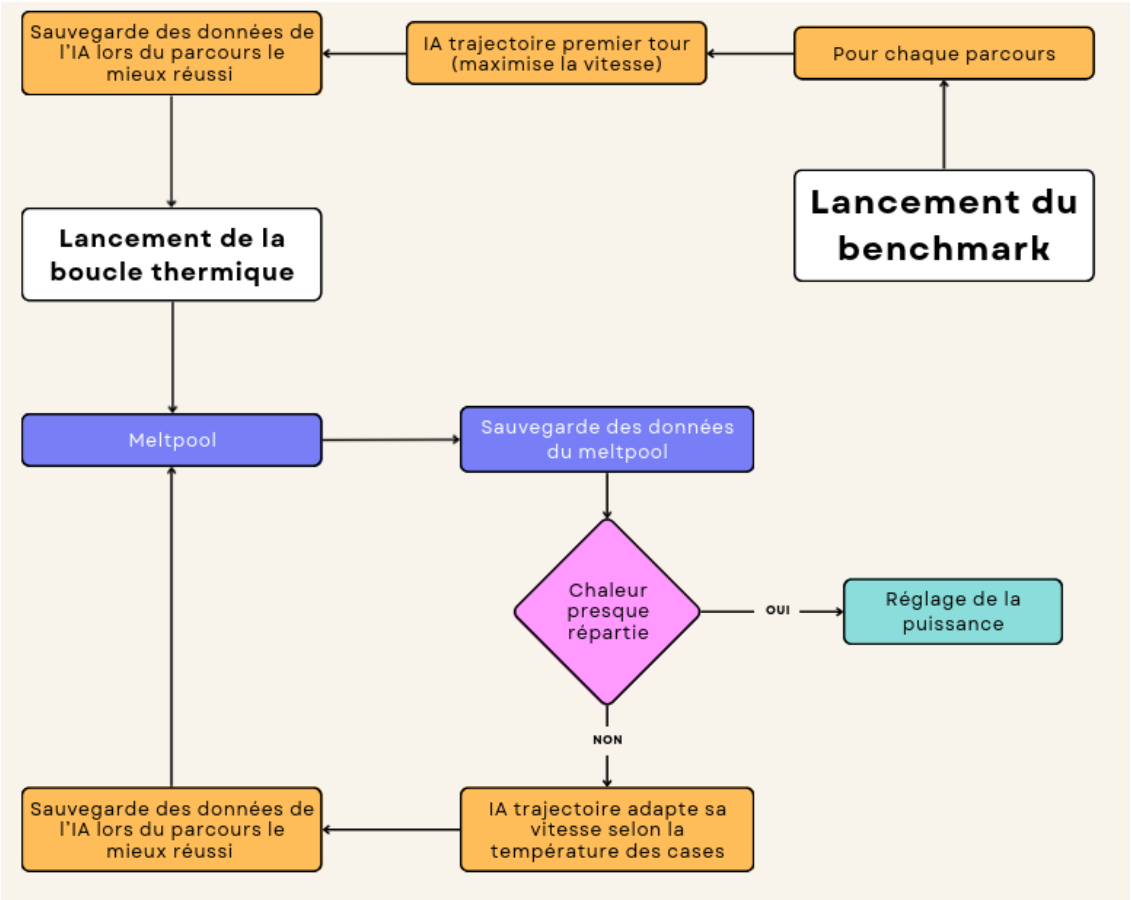

Figure 6: Boucle d'apprentissage automatique incluant la partie thermique

3 Impacts sociétaux et environnementaux

Lors de mon stage, j'ai donc effectué des travaux informatiques, me rendant tous les jours au lieu de travail. De ce fait, mon activité à eu un impact sur l'environnement qui peut être compris en deux points. Tout d'abord, ce sont mes déplacements qui ont eu le plus d'impact sur mon bilan carbone durant ce stage. Nous pouvons voir que mes trajets entre mon domicile et le lieu de travail ont été variés; ceux-ci sont détaillés ci-dessous, résultant en un total de 18.4kg CO_2e:

- Lors de la première semaine de stage, mon moyen de transport était le bus. Allant au laboraotire de l'I2M le matin pour revenir à mon domicile le soir. Le trajet étant de 3.1km, celui-ci étant répété 2 fois par jour pour une durée d'une semaine de travail, nous trouvons que j'ai parcourus 31 kilomètres en bus. Selon l'Ademe[6], l'impact carbone d'un kikomètre en bus est de 113g CO_2e. Pour la distance totale que j'ai parcourue, nous obtenons environ 3.5 kg CO_2e.

- Lors du reste du stage, j'ai utilisé une trottinette électrique afin de me rendre au lieu de stage. Rentrant la plupart des jours à mon domicile entre midi, je répétais environ 4 fois un trajet de 2.5km par jour, soit un total de 10km par jour. M'étant rendu précisément 60 jours au lieu de travail en trottinette élecrtique, cela fait un total de 600km. Selon l'Ademe, l'impact carbone correspondant est de $24.9 \times 600 = 14940$g CO_2e, soit 14.9 kg CO_2e.

[6]https://agirpourlatransition.ademe.fr/particuliers/bureau/deplacements/calculer-emissions-carbone-trajets

Nous observerons désormais l'impact du travail que j'ai effectué. On ne relève ici qu'une source de production de CO_2e, qui est dûe à l'ordinateur portable que j'ai utilisé durant le stage. En se basant sur Ecodiag[7], nous pouvons voir que le modèle d'ordinateur portable que j'ai utilisé, soit un Dell Latitude 5490, produit 75kg CO_2e/an en présumant que sa durée de vie est de 4 ans. Si on considère seulement les jours travaillés, qui sont au nombre de 66, il devrait être possible de plus précisément estimer mon impact. Grâce au calcul suivant, $66 \div 365 \approx 0.18$, on peut estimer que ma production de CO_2e lors de ce stage correspondrait à 18% de 75Kg CO_2e, soit à 13.5kg CO_2e environ.

Au total, mon bilan carbone lors de ce stage à été d'environ 31.9kg CO_2e. Il est aussi intéréssant de noter que mon utilisation de la trottinette électrique a eu pour effet de réduire la production de CO_2e de moitié concernant les transports. Malgré avoir preque doublé la distance journalière parcourue par 2, la production de CO_2e de la trottinette correspondant à un quart ce celle du bus, on retrouve effectivement un impact réduit par 2. En guise d'exemple, le calcul $1 \times 2 \div 4 = 0.5$ permet d'expliquer mes propos, où *2* correspond à la distance doublée et *4* correspond à la production de CO_2e divisée par 4.

On remarque cependant un axe d'amélioration évident sur ce point. Si j'avais pris la décision de manger sur place, j'aurais pu réduire ma consommation de transport par 4 au lieu de seulement 2 lors de mon utilisation de la trottinette. J'aurais même pu de choisir la marche pour me rendre au lieu de travail, ce qui aurait été faisable étant donné qu'il n'est situé qu'à 2.5km de mon domicile.

4 Bilan

Au cours de ce projet, j'ai donc pu développer une IA d'apprentissage par renforcement profond, prenant en compte divers aspects du procédé LPBF afin que celle-ci le représente au mieux. Cela m'a permis de développer mes compétences en Deep Learning et en Reinforcement Learning, ainsi qu'en Data Analysis. Plus précisément, du point de vue de l'IA, j'ai pu revoir plus en détails les bases théoriques derrière son fonctionnement, que j'avais pu voir une première fois à l'IUT, mais ce projet m'a surtout permis de vraiment comprendre le fonctionnement de l'apprentissage par renforcement, notamment du point de vue des états et récompenses. Concernant l'analyse de données, j'ai pu comprendre l'importance de la phase de traitement de données, avec les techniques associées tout en apprenant les divers moyens qui existent pour en tirer des informations, comme quel graphique il faudrait faire, avec quelles données et dans quel but.

Par rapport au projet en lui-même, on peut voir un certain nombre d'axes d'amélioration. D'une part, l'environnement ne possède actuellement que 3 vitesses et 7 directions, et toutes les positions sont sous formes de cases dans une grille. On pourrait améliorer ça en utilisant des positions flottantes, qu'on pourrait potentiellement approximer aux cases, et avec en même temps des vitesses elles aussi sous forme de nombres à virgule. Cela permettrait de mieux représenter le procédé, et d'avoir plus de moyens d'affiner l'apprentissage et d'optimiser les résultats. On pourrait aussi corriger le fonctionnement des mouvements diagonaux. Avec l'utilisation des cases, le serpent en parcours autant que cela soit en ligne droite ou en diagonale. Cependant, si on prend exemple sur un triangle rectangle isocèle, on remarque que la longueur de hypoténuse est supérieur aux autres. Si on superpose ce triangle sur une case, on comprend que le laser ne serait pas supposé parcourir autant de cases en ligne droite qu'en diagonale, la distance étant effectivement plus longue. Autrement dit, la longueur de 3 cases en diagonales est plus grande que celle de 3 cases ligne droite, ce qui n'a pas été pris en compte dans le code actuel.

D'un point de vue personnel, ce projet me servira énormément pour la suite de mes études et pour mon projet professionel, étant donné que je souhaite me diriger vers le Machine Learning ou vers la science des données. J'ai en effet beaucoup apprécier les missions qui y sont associées, et que j'ai pu réaliser durant le stage. Travailler dans l'IA était déjà mon but initial, mais je m'imagine désormais tout aussi bien ou presque travailler dans le domaine des données.

[7] https://ecoinfo.cnrs.fr/ecodiag-calcul/

Annexes

hugo.martins5733@gmail.com

github.com/Hugo33martins

07 61 72 36 15

GRADIGNAN (33)

Martins Hugo

OBJECTIF

INGENIEUR MACHINE LEARNING

PROFIL

Je suis étudiant de 3ème année de BUT Informatique à l'IUT de Bordeaux. Engagé et persévérant, j'ai réalisé divers projets en équipe et en autonomie. J'aimerai devenir Ingénieur en Machine Learning, car j'ai toujours été intéressé par ce qui relève de l'IA. Je trouve en particulier que ses applications dans le secteur de la santé et dans l'industrie sont passionants.

COMPÉTENCES

Langages de programmation:
- Python
- Java
- C#
- C/C++

Développement web:
- PHP (Symfony)
- Javascript (React, TS)

Base de données:
- SQL (Oracle, MSSQL)
- NoSQL (MongoDB)

Virtualisation:
- Docker

Gestion de projet / collaboration:
- Méthode agile (SCRUM)
- GIT

CENTRES D'INTÉRÊTS

- Japon
- Jeux vidéos

EXPERIENCE

Avril à Juin 2024

UNIVERSITÉ DE TSUKUBA JAPON

- Analyse des données d'un dataset à l'aide d'un Random Forest ML model
- Étude des features du dataset: data visualization et SHAP values
- Rapport et présentation LaTeX

Février à Mai 2025

LABORATOIRE DE L'I2M UNIVERSITÉ DE BORDEAUX

- Développement d'une IA basée sur l'apprentissage par renforcement profond (modèle DQN avec PyTorch)
- Analyse des données et résultats de l'IA pour l'optimiser et expliquer les observations (via Python et les librairies associés à la Data Science)

PARCOURS

De 2022 à 2025

BUT Informatique

I.U.T INFORMATIQUE DE BORDEAUX

- Parcours A : réalisation d'applications
- Parcours international (anglais) Score TOEIC (juin 2024) : 950/990, équivalent niveau C1

De 2019 à 2022

Baccalauréat général - mention bien

LYCÉE GÉNÉRAL ET TECHNOLOGIQUE JEAN-MOULIN

- Section européenne
- Spécialités maths et informatique

EMPLOI

Juin à Juillet 2023

CHATEAU DU HAYOT

Ouvrier viticole