Rapport intermédiaire pour Projet Long
Date de rendu : 28 février 2023

Alban LE JEUNE, Bilal SEDDIKI

28 février 2023

1 Introduction générale et contexte

Le monde des échecs a été frappé récemment avec un scandale de triche sans précédent. Le 4 septembre 2022, lors de la coupe Sinquefield, Magnus Carlsen, actuel champion du monde, perd avec les blancs en partie classique. Pour un néophyte, cela ne semble pas surprenant d'être amené à perdre lors d'un tournoi. Sauf que pour cet homme, cela n'était pas arrivé depuis octobre 2020, contre le Grand Maître Levon Aronian.
Suite à cette défaite, Magnus Carlsen quitte le tournoi et insinue que son adversaire, le jeune Grand Maître américain Hans Niemann, aurait triché, d'une façon ou d'une autre. Cette accusation est le point de départ d'une enquête de la *FIDE* (Fédération Internationale des Échecs) qui va durer plusieurs mois ; et qui est toujours en cours. La théorie soulevée par cette dernière est la suivante : Niemann aurait eu un complice qui lui aurait transmis le coup le plus intéressant lors de moments critiques de la partie, durant le *mid-game* (milieu de partie). Le moyen de communication aurait été un appareil discret dissimulé retranscrivant les coups en vibrant.

Notre projet consistera à reproduire la théorie émise par l'organisation, à la différence près qu'il sera alimenté par la puissance du *machine-learning* et de l'intelligence artificielle. Notre objectif est le suivant : un *parser* d'image de partie d'échec, dont l'analyse sera couplé à *Stockfish*, l'intelligence artificielle d'échec la plus performante existante, qui transmettra le résultat de ses calculs sur un appareil discret. La transmission sera effectué avec une série de vibrations permettant d'indiquer à l'utilisateur une des 64 cases de l'échiquier, en utilisant la notation classique (A1, B2, C3, etc.). La communication est possible grâce à un *Arduino* connecté à un moteur et recevant les informations via *bluetooth* ou par Internet. Le *parser* fonctionnera entre autre via du *machine-learning* entraîné avec un *data-set* fournis de photos afin de lui permettre d'apprendre à distinguer les différentes pièces présentes dans un échiquier. Le *parser* a lui seul ouvre la porte a plusieurs fonctionnalités externes, comme l'exportation d'une partie réelle à une partie numérique via *chesslib*, à titre d'exemple.

2 Avancement du projet

Le projet jusqu'ici avance bien, les délais que nous nous étions imposés sont en grande partie respectés et il n'y a pas de réel impasse quant à la progression de ce dernier.

2.1 *Parsing* d'image

Le perfectionnement du *parsing* est notre priorité courante, afin de garantir une exacte valeur d'entrée afin que l'intelligence artificielle de calcul de position puisse agir correctement. À l'heure actuelle, les images telles que celle représentée dans la Figure 1 parviennent à être partiellement analysées : notre projet en tire les coordonnées des cases et arrive à détecter avec un taux élevé de réussite la présence d'une pièce dans une case. Des tests unitaires simples ont été créé avec comme source une banque d'images avec différentes lumières, tailles et formes d'échiquiers. Le *parser* pâtit toutefois toujours pour la reconnaissance des pièces. Nous avons pris la décision d'opter d'un type d'entrée alternatif, dont nous parlons plus en détail dans la section Prochaines étapes de développement. En d'autres termes, l'analyseur pour photo prise en vue de haut peut être considéré comme complété, mais l'analyseur d'image en lui-même n'est pas encore au point.

2.2 Intelligence artificielle

Stockfish est bien implémenté, paramétrable en termes de temps et de profondeur de recherches afin de nous fournir une solution optimale à n'importe quel échiquier que nous lui envoyons. Un générateur de position aléatoire simple a également été implémenté pour le tester et calibrer l'intelligence artificielle selon ce qui nous semblait être le meilleur compromis entre efficacité et rapidité. Nous sommes capable de déterminer si une position testée est possible dans une partie d'échec, afin d'éviter tout test inutile. Si le *parsing* n'est pas concluant, nous pouvons également fournir la *FEN* (*Forsyth–Edwards Notation*, notation standard des parties d'échecs) à *Stockfish* afin qu'il puisse également calculer le prochain meilleur coup. Cette partie du projet peut être considéré comme complète.

FIGURE 1 – Un exemple d'image de test pour le parser

3 Difficultés rencontrées jusque là : le parsing d'image

3.1 Plan original

À l'origine, le *parsing* d'images d'échiquier que nous envisagions fonctionnait principalement avec pour entrée des photos prises en vue de dessus.

3.2 Des avantages clairs et une apparence simple...

Ce type d'image présente plusieurs avantages significatifs à premier coup d'oeil :

— l'obtention relativement simple des coordonnées des cases
— la couleur des cases et de la pièce qu'elle contient
— la reconnaissance simple de la présence ou l'absence d'une pièce dans une case. Bien que non-nécessaire étant que le processus de reconnaissance de la pièce peut déterminer cette information, disposer d'une méthode complémentaire pour augmenter la précision de nos résultats est très utile

Ces arguments nous étaient suffisants pour lancer le développement avec ce format d'image tête baissée. Pendant plusieurs mois, ce fut le seul format de photographie supporté par notre code.

3.3 ... camouflant une certaine complexité

Bien que visiblement simple, cette méthode s'est révélé néanmoins complexe à écrire :

— la récupération des coordonnées des cases nécessite une retouche assez lourd de l'image : luminosité et contraste augmenté, transformation des couleurs de l'image en niveaux de gris, flou gaussien, etc.
— déterminer si une case était vide relevait de l'analyse de l'histogramme de l'image de la case : un histogramme avec peu de sommets signifiant une case vide et l'inverse une case avec une pièce.

En d'autres termes, cette analyse à l'apparence simple requiert un réglage fin lourd. En effet, le niveau d'augmentation de luminosité et de contraste et le niveau de flou idéal dépend de chaque image, ce qui détermine un pic dans un histogramme dépend du seuil que l'on choisit n'est également pas chose aisée. Trouver une valeur qui satisfait la majorité des cas à demander plusieurs longs et vigoureux tests et fut fastidieux. Cette mission fut toutefois accomplie.

3.4 Les limites de la simplicité

Une fois les premières étapes de parsing de l'échiquier cité ci-dessus réalisés, le problème fut le suivant : il est en réalité assez difficile de reconnaître une pièce d'échec avec comme unique source une photo prise en vue de haut, et ce même pour un humain. Le contexte de l'image peut aider :

— règles du jeu qui rendrait certaines suppositions illégales
— comparaison à des pièces de la même image

Cela dit, ces éléments ne suffisent pas à eux-mêmes dans la plupart des cas. Ajoutons à cela la diversité de formes que les pièces peuvent prendre, les distinguer avec un taux décent de succès semblait mission impossible, même via *machine-learning*, à moins d'avoir entraîné cette dernière spécifiquement avec les pièces en question. Cette simple impossibilité nous force à adapter notre base de code et a définitivement été le plus gros obstacle face auquel nous avons pu faire face depuis le début de ce projet. Nous avons cependant un autre format d'image en tête dont nous parlons dans Prochaines étapes de développement.

4 Prochaines étapes de développement

La suite de développement va tourner autour de l'*Arduino* et du perfectionnement de l'intelligence artificielle pour le *parsing*. Les pistes à suivre pourront consister en l'augmentation de la taille du *data-set*, l'amélioration de l'algorithme actuel, trouver un modèle plus performant.

4.1 Terminer le parsing d'image

Nous opterons, comme image d'entrée, des photos d'échiquiers prises en vue de haut avec un léger angle, comme illustré dans la figure 2. Cela nécessitera une réécriture partielle de notre implémentation actuelle de notre *parser*, mais fort heureusement, notre travail passé ces derniers mois sur le *parser* pour photo prise en vue de haut devrait nous servir.

4.2 Entamer la couche physique

La partie gestion de logique des échecs et Stockfish étant terminé, la partie *Arduino*/électronique va pouvoir commencer. Cette partie va être une complète découverte car aucun de nous deux n'avons déjà manipulé un tel outil. La réalisation d'un objet technique est également un nouveau challenge pour nous. Même si nous avons des idées relativement précises sur ce que l'on souhaite avoir comme résultat, y arriver représente un défi. Beaucoup de questions subsistent quant à la forme que prendra cet objet, trouver quelque chose correspondant à notre cahier des charges (discrétion, faible poids, électronique non apparente au maximum, etc.).

4.3 Front-end utilisateur

Afin de faciliter au plus l'usage de notre logiciel, une application sera développée afin de ne pas avoir à passer par la nécessité d'avoir un ordinateur sur place et de pousser encore plus loin la discrétion du dispositif. Cette application permettrait au complice d'agir plus facilement en ajoutant les coups suivants au *FEN* pour accélérer le processus ou bien de faire directement fonctionner le *parser* via l'appareil photo du téléphone. Si le temps nous le permet, avoir une interface graphique d'un échiquier pour simplifier ses actions.

FIGURE 2 – Exemple de photo d'échiquier prise en vue de haut avec un angle