Projet IA

Rapport

Perret Quentin
Roche Eléa

A l'Ecole Nationale Supérieure de Cognitique

BORDEAUX INP Ensc

Sommaire :

Sujet

Le sujet choisi pour notre projet de parcours IA est le sujet numéro 1, soit “Implémentation d’une IA dans un contexte ludique”.
Nous avons donc décidé d'appliquer un algorithme d’IA dans le jeu osu!, afin que l’IA s’entraîne pour battre des records.

Description du jeu

Osu! est un jeu de rythme dans lequel le joueur doit accomplir des actions en rythme avec la musique.
Le but du jeu est de cliquer sur les objets en rythme, de manière à marquer le plus de points possible et d'obtenir le plus grand score possible afin de gagner des points de performance, représentant le niveau du joueur. Ces derniers sont classés selon ce score. Plus le joueur respecte le rythme de la beatmap, plus il obtient un grand score de précision et donc de points, deux paramètres qui influent directement sur le gain des points de performance.
À chaque fin de beatmap, une note est donnée au joueur selon la précision et le combo obtenu allant de D pour la pire à SS pour une réussite parfaite (D, C, B, A, S et SS).

Le joueur sait sur quel objet cliquer grâce aux numéros (il doit cliquer sur les objets par ordre croissant). De plus, il sait quand cliquer, à la fois grâce à la musique (car le jeu est en rythme) mais aussi grâce aux formes. En effet, peu de temps avant que le joueur doit cliquer sur un objet, un cercle apparaît autour de l’objet en question. Ce cercle rétréci progressivement jusqu’à se superposer aux contours de l’objet. Plus le joueur clique au moment où la superposition à lieu, plus le joueur gagne de point.
Nous avons donc décidé de ne pas nous appuyer sur la musique pour que le joueur sache quand cliquer, mais plutôt sur la détection des formes et des contours.

Certains objets ont un comportement particulier :

- les sliders : le joueur doit cliquer/taper sur le début du slider, puis, en maintenant le bouton enfoncé, suivre une boule mobile (appelée slider ball) le long du chemin jusqu'à ce que la slidertail soit atteinte.
- les spinners : pour compléter un spinner, le joueur doit maintenir le bouton de la souris ou du clavier enfoncé. À partir de là, il doit utiliser la souris et faire tourner le

spinner (dans un des deux sens) jusqu'à ce que le cercle du spinner s'agrandisse complètement.

Objectif métier

Pour réaliser ce projet, nous allons donc appliquer un algorithme d'IA, plus précisément de Reinforcement Learning, afin que le joueur fictif apprenne à gagner des points et à réaliser le meilleur score et la meilleure précision. Notre objectif est donc que l'algorithme apprenne en essayant de maximiser le score et la précision qu'il réalise au cours d'une partie.

Environnement technique

Notre projet est développé en langage Python, les algorithmes de machine learning utilisent le framework PyTorch. De plus, la gestion des dépendances est faite avec Poetry. Un versionning est mis en place à l'aide de Git et un dépôt public sera disponible tout au long du projet sur Github au lien suivant : https://github.com/elroche/IA_oso_player.

Architecture du code

Récupération des des images

Ayant fait le choix de ne pas reproduire ou de ne pas utiliser une reproduction du jeu osu!, il était impossible pour nous d'obtenir " l'état " du jeu: i.e les éléments présent sur l'écran ainsi que d'autres informations plus ou moins utiles comme la vie du joueur, son score, sa précision ... .

Afin d'utiliser le jeu original il est donc nécessaire de récupérer directement l'écran visible du joueur en réalisant des screenshots. Après avoir réalisé un benchmark sur la vitesse de réalisation de screenshot notre choix s'est porté sur la librairie PIL et sa classe ImageGrab. Les screenshot sont réalisé en noir et blanc afin de limiter la place qu'il occupe dans la mémoire.

Enfin une fois un screenshot fait celui-ci dans une queue. Ces images sont ensuite utilisées pour obtenir la précision du joueur à l'aide du modèle tesseractOCR et pour être dans notre modèle entraîné par renforcement un fois prétraité et formaté à l'aide un CNN.

Principe du multithreading

Comme expliqué précédemment, notre projet n'utilise pas une reproduction du jeu Osu! et s'appuie donc sur des screenshots réalisés à partir de l'écran du joueur. Afin de pourvoir réaliser en simultanée les différentes actions nécessaires au bon fonctionnement du robot, il a été nécessaire de mettre en place du threading (thread[1] : exécution d'un ensemble d'instructions du langage machine d'un processeur).
Deux actions principales doivent être réalisé :

- la récupération des images et leur stockage dans une queue comme expliqué précédemment.
- la réalisation d'inférence par le modèle entraîné afin de lui permettre de tester et d'explorer l'environnement de jeu.

L'avancement du projet n'a pas permis de déterminer si la mise en place d'un troisième thread permettant de juger, noter et corriger le modèle était nécessaire dans notre cas.

Les images sont donc communiquées à l'aide d'une file d'attente, une structure de données qui suit le principe du premier entré, premier sorti (FIFO). Dans ce contexte de programmation multithread, une queue est utilisée pour la communication entre threads et permet donc de transmettre les images vers les différents modèles ou programmes les utilisant.

Modèle de tesseract

Ce modèle a pour objectif de détecter le score et la précision. En effet, ce sont ces données dans le jeu qui permettent au joueur d'évaluer ses performances.

Le score est un nombre, qui augmente ou diminue en fonction de la réalisation des actions. Si le joueur réalise une bonne action, son score augmente, alors que s'il ne réalise pas l'action, son score diminue. Le score n'évalue pas la précision de la réalisation de l'action.

La précision est une mesure en pourcentage, qui évalue la capacité d'un joueur à toucher les objets à temps. Plus le joueur est précis dans la réalisation de l'action au bon moment, plus la précision augmente. Au contraire, moins il est précis, plus sa précision diminue.

Ces deux valeurs sont donc indispensables à notre projet, permettant au programme de savoir s'il réalise les bonnes actions où non, et d'améliorer sa précision.

Afin de récupérer ces données, nous utilisons Tesseract, qui est un moteur OCR (Optical Character Recognition) open-source qui extrait le texte imprimé.

Notre tesseract model permet ainsi de détecter s'il y a des nombres ou non en haut à droite de l'écran, et s'il y en a, d'extraire ces données. Il nous permet ainsi de récupérer le score et la précision lors d'une partie de jeu.

Cependant, nous avons fait le choix de ne garder que la précision, et de ne pas garder le score. En effet, tesseract permet de faire detection d'un groupe de caractère. Si nous gardons l'extraction de ces deux données, cela nécessite un traitement supplémentaire afin de séparer les deux données pour obtenir le score et la précision. Cela ralentit donc le modèle, en passant de 2 secondes par image pour une détection des 2 valeurs, à environ 80 millisecondes par image en ne récupérant que la précision.

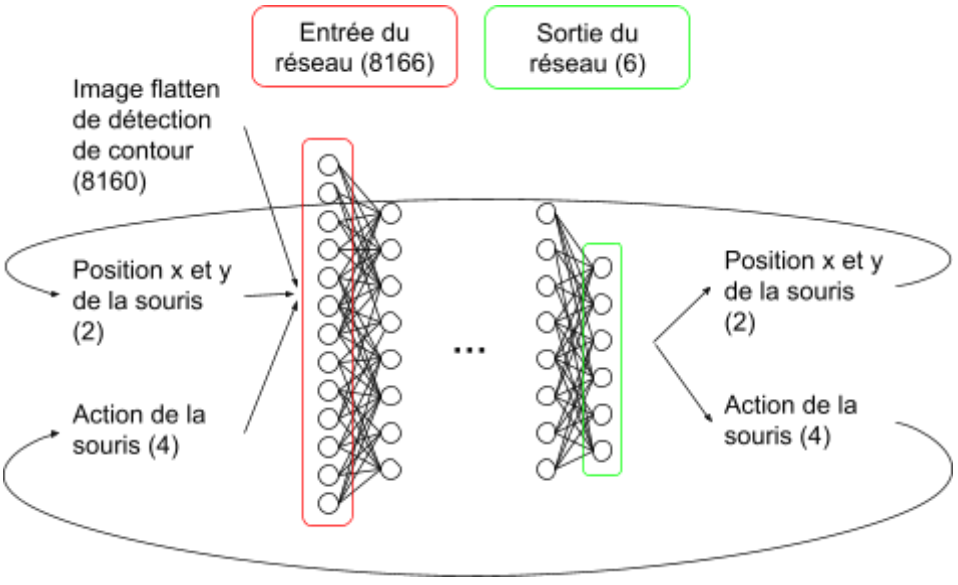
C'est ainsi que pour une question d'efficacité du modèle, nous avons fait le choix de ne garder que la précision, qui nous apporte des données suffisantes pour connaître l'état du jeu.

Modèle de prétraitement d'image

Ce modèle permet de réaliser la détection de contours sur des images, grâce à l'utilisation d'un CNN (convolutional neural network) et notamment l'application de plusieurs couches de Max Pooling. En effet, il sera plus facile pour notre intelligence artificielle d'apprendre et de fonctionner sur des images moins volumineuses, n'ayant pas d'information inutile dans son apprentissage.

Modèle de renforcement

Ce modèle a pour objectif de réaliser l'apprentissage par renforcement sur les données d'entrée.



Pour cela, il est composé de plusieurs couches de neurones. Il prend en entrée le dernier screenshot de sa liste, l'action de la souris sous forme de vecteur[2] (clic simple, clique continue, clic lâché ou rien) ainsi que la position de la souris (x et y) et ressort la prochaine action à réaliser avec les positions de la souris.

Avant de rentrer dans le modèle, chaque image est d'abord envoyée dans le modèle de prétraitement d'image afin de ne traiter que les images allégées.

Initialement, nous souhaitions avoir en première couche de notre réseau de neurone un LSTM (Long Short-Term Memory) car ce type de réseau de neurone est conçu pour mieux gérer les dépendances à long terme dans les données. Ils sont particulièrement efficaces pour traiter des séquences temporelles ou des données séquentielles, car ils peuvent retenir des informations importantes sur de longues périodes. Le jeu Osu! étant un jeu de rythme, il y a donc une cohérence temporelle, un LSTM aurait donc été adapté.

Cependant, après de nombreuses tentatives d'intégration d'un LSTM dans notre projet, nous avons dû abandonner cette idée car la sortie du réseau restait toujours constante.

Gestion de projet

Afin de progresser efficacement sur l'ensemble des tâches à réaliser pour ce projet, 2 axes ont été identifié et travaillé en parallèle par les membres du groupe:

- Une approche, réalisée par Quentin, concernait la mise en place de la pipeline de récupération d'image et la mise en place et maintenance de la programmation multithread.
- L'autre approche, réalisée par Eléa, concernait la définition et l'entraînement des différents modèles permettant d'extraire les informations nécessaires pour l'apprentissage par renforcement.

Perspectives et améliorations possibles

Intégration d'un LSTM

Afin d'améliorer notre modèle, il serait intéressant de réussir à y intégrer et mesurer l'impact d'un LSTM ou autre format de couche récurrente, qui comme expliqué précédemment, serait adapté dans le cadre de notre projet en prenant en compte les séquences temporelles.

Finalisation de l'apprentissage par renforcement

De plus, il faudrait adapter nos programmes pour intégrer un modèle de Deep-Q learning, une technique d'apprentissage par renforcement profond utilisée pour résoudre des problèmes complexes, en particulier dans le domaine des jeux vidéo et de la prise de décision.

Entraînement et modification du modèle

Pour finir, il faudrait par la suite réaliser un entraînement du modèle, et en fonction des résultats obtenus, l'ajuster si nécessaire, afin d'obtenir une meilleure précision et de meilleurs résultats.

Conclusion

Pour conclure, nous pouvons dire que ce projet nous a permis d'enrichir nos connaissances ainsi que nos compétences dans le développement d'une intelligence artificielle.

En effet, nous avions déjà étudié à quelques reprises l'apprentissage supervisé, mais c'était la première fois que nous nous lançons dans le développement d'un apprentissage par renforcement.

De plus, il nous a aussi permit de comprendre le fonctionnement et l'utilité des environnements virtuels et de Poetry.

Enfin, il nous a permis de comprendre l'importance et la difficulté de la récupération et le pré-traitement des données afin que ces dernières soient exploitables dans l'élaboration d'une intelligence artificielle.