TetrisAI.jl: Apprentissage par renforcement appliqué à Tetris (EQ13)

Léo Chartrand Jeremi Levesque Olivier Thiffault

Université de Sherbrooke

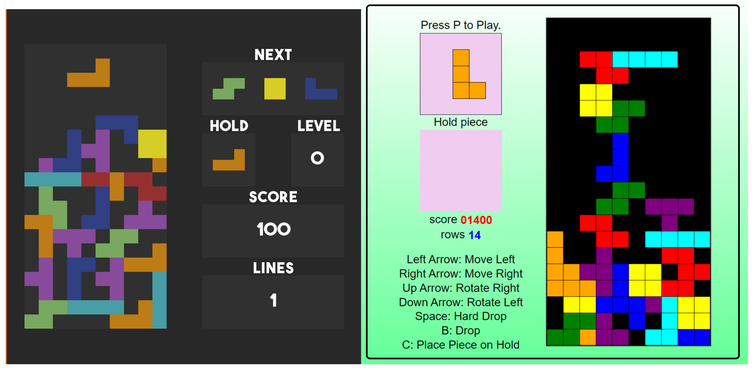
Résumé

TetrisAI.jl est un logiciel écrit en Julia qui permet de tester la performance de différents algorithmes d'apprentissage machine profond sur l'environnement Tetris conforme à la version NES. Des algorithmes de pointes en apprentissage en renforcement apprennent lentement même avec une modélisation des récompenses et une extraction de caractéristiques. Notre module d'apprentissage par imitation permet à nos agents d'apprendre initialement à reproduire les mouvements faits par des joueurs experts grâce à notre banque de données générée par notre plateforme de collecte de données expertes.

TetrisAI.jl: NES Tetris

TetrisAI.jl est constitué d'une représentation interne du jeu de Tetris ainsi que d'une interface graphique permettant aux humains de visualiser et d'interagir avec le jeu. Le logiciel offre plusieurs options à l'utilisateur:

1. **Tetris classique** – Il est possible pour un humain de jouer à Tetris avec une interface graphique (Voir Fig. 1a)
2. **Démo de modèle** – Permet de visualiser les comportements d'un IA en lui faisant jouer à Tetris avec affichage de l'interface graphique
3. **Génération de données d'entraînement** – Permet à un humain de jouer à Tetris et d'enregistrer ses actions pour générer des données d'entraînement pour l'apprentissage supervisé
4. **Entraînement d'un agent** – Un agent donné s'entraîne à jouer à Tetris sur un certain nombre de parties, sans l'affichage d'une interface graphique



(a) Interface de jeu en Julia **(b)** Interface web

Figure 1. Interface utilisateur en Julia pour la démonstration et pour jouer. L'interface web a été créée pour faciliter la récolte de données expertes pour le pré-entraînement d'un agent.

Tetris possède plus de 2^{200} configurations de grilles possibles ce qui pose un défi de complexité intéressant pour l'entraînement d'un agent. Certaines recherches simplifient le jeu afin de réduire l'espace d'états et/ou l'espace d'actions d'un agent afin d'obtenir des performances supérieures. Cependant, notre implémentation cherche à entraîner un agent avec les mêmes actions et un environnement fidèle à celui de la NES.

Apprentissage machine

- **Simulation de l'apprentissage humain** – Nous pouvons lancer un agent dans un environnement d'apprentissage sans lui introduire les concepts et les règles de bases du jeu, et il saura optimiser son jeu en fonction de sa performance.
- **Apprentissage par renforcement** – Des algorithmes de pointe, soit PPO et DQN, qui sont généralement des algorithmes de choix dans le domaine de l'apprentissage par renforcement sont disponibles pour l'entraînement d'agents.
- **Apprentissage par démontration** – Permet à l'agent de s'entraîner sur des données étiquettées générées au préalable par un expert du domaine afin de guider l'agent vers la complétion de quelques lignes afin que les algorithmes d'apprentissage par renforcement puissent paufiner une stratégie unique en apprenant sur des récompenses représentatives.
- **Reproducibilité des résultats** – L'ensemble de données expertes pour le pré-entraînement sera disponible pour tous dans un format commun et facile d'utilisation afin que tous puissent reproduire nos résultats.
- **Structure modulaire** – Comme ce framework sera ouvert publiquement, sa structure repensée permettra de facilement ajouter et tester de nouveaux algorithmes d'apprentissages grâce à la réutilisation de code et au système d'étalonnage des performances.

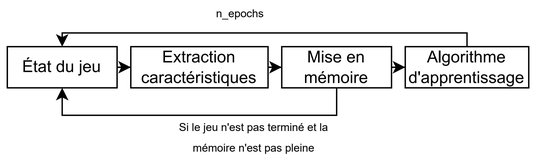


Figure 2. Architecture d'apprentissage

Extraction de caractéristiques

Puisque l'environnement possède un très grand espace d'états, nous avons tenté d'ajouter les fonctionnalités suivantes pour pouvoir guider l'agent vers l'obtention de ses premières récompenses.

- **Extraction de caractéristiques pertinentes** – Permet de complémenter l'espace d'états envoyés aux réseaux neuronaux de l'agent en identifiant des caractéristiques pertinentes par rapport à l'état de la grille (i.e. trous accessibles, trous inaccessibles, hauteur moyenne, la variation de hauteur).
- **Modelage des récompenses** – On utilise un sous-ensemble des caractéristiques pertinentes pour calculer des récompenses intermédiaires pour inciter un placement de pièces idéal. Les récompenses sont donc calculées entre la différence de score intermédiaire entre deux ticks d'un agent où ce score est calculé de la façon suivante:

  $(-0.51 \times height_avg) + (0.76 \times lines) + (-0.35 \times holes) + (-0.18 \times bumpiness)$
- **Réseaux neuronaux à convolution** – Des réseaux à convolutions d'environ 420k paramètres permettent de simplifier l'espace d'états de la grille en ayant extraire des caractéristiques déterminées par un réseau de neurones.

Résultats & analyse

Dans le cadre de ce projet, nous avons continué le développpement de la plateforme TetrisAI.jl, entièrement écrite en Julia pour permettre l'implémentation et l'étalonnage d'algorithmes d'apprentissage par renforcement (RL). De plus, nous avons mis sur pied un cadre pour accélerer l'entraînement d'agent avec de l'apprentissage supervisé afin de palier à la difficulté d'obtenir les premières lignes complétées.

Comme nous pouvons le constater dans la figure 3, les agents font toujours face à un plateau. Les algorithmes de type *Markov Decision Process* à eux seul ne semblent pas suffisants pour pouvoir exploiter un environnement aussi complexe. À travers la littérature et nos expériences, l'environnement offre des récompenses beaucoup trop clairsemées pour qu'un agent puisse apprendre rapidement. L'agent doit effectuer une longue séquence d'actions avant de pouvoir avoir un retour (une récompense négative ou positive) sur les actions qu'il prend. Si une longue suite d'action mène à une récompense positive, doit-il privilégier toutes les actions de cette séquence ou certaines d'entre-elles ne sont pas optimales?

Concours incitatif

Afin de récolter des données expertes, nous avons préparé un concours afin d'inciter les gens à jouer sur notre plateforme web où le meilleur joueur gagnera une carte cadeau SAQ d'une valeur de 40$. Ce concours nous a permis de récolter environ 600 parties valides pour un ensemble de pré-entraînement

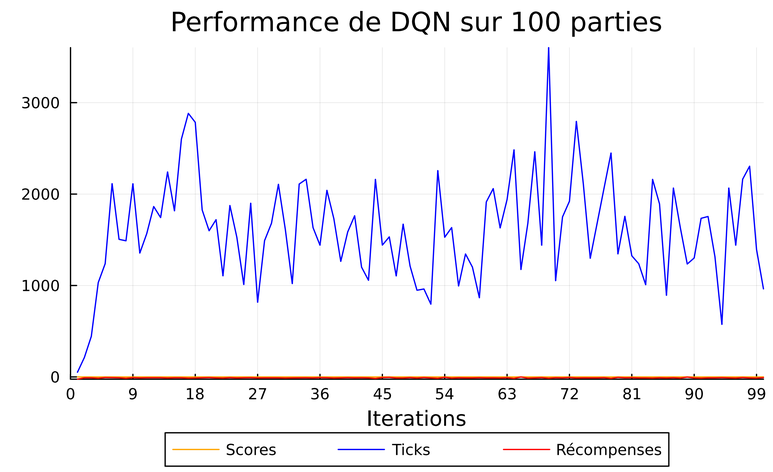


Figure 3. Performances des techniques d'apprentissage

Les améliorations proposées

- Augmentation du temps d'entraînement (grape de calcul parallèle)
- Algorithmes de recherche heuristique (ex. Monte-Carlo Tree Search)
- Recherche d'hyperparamètres

Nous remercions la supervision du Professeur Djemel Ziou, PhD. pour nous avoir guidé tout au long de ce travail.