Projet Deep Learning

Groupe 3

PPO – Proximal Policy Optimization (1)

- RL : pas de dataset statique (l'agent génère son propre dataset en interagissant avec son environnement) → la distribution des récompenses et des observations changent constamment
- RL : très sensibles aux hyperparamètres

- ➔ OpenAi met au point l'algorithme de **PPO**
  - Policy Gradient method (différent de Q learning) – pas de replay buffer

  C-a-d utilise une seule fois les expériences collectées

- Descente de gradient sur un seul batch → risque de mise à jour des trop drastique paramètres

- PPO basée sur Trust Region Policy Optimization (TRPO)
  - TRPO dans cette politique une contrainte est ajoutée (KL) pour que l'actualisation de la politique ne soit pas trop importante
  - PPO intègre cette contrainte directement dans sa politique

Source : arXiv:1707.06347

PPO – Proximal Policy Optimization (2)

Fonction de coût :

$$L^{PPO} = \widehat{E}_t\,[\,min(r_t(\theta)\widehat{A}_t, clip(rt(\theta), 1-\epsilon, 1+\epsilon)\,\widehat{A}_t) - c_1 L^{VF}(\theta) + \beta S[\pi\theta](st)\,]$$

Espérance [min entre (Ratio proba de l'action a_t suivant la nouvelle sur ancienne politique * Estimation de la récompense de l'action choisie et Ratio proba de l'action a_t suivant la nouvelle sur ancienne politique * Estimation de la récompense de l'action choisie bornée dans l'intervalle $[1-\epsilon, 1+\epsilon]$) - Terme pour actualiser la base line + Terme ajoutant de l'entropie pour que l'agent explore assez]

min entre (Ratio * (Reward obtenue - ligne de base) et Ratio * (Reward obtenue - ligne de base))

min entre (Ratio * (est-ce que l'agent a fait mieux ou moins bien que prévu) et Ratio * (est-ce que l'agent a fait mieux ou moins bien que prévu))

✓ >0 ✗ <0

Source : arXiv:1707.06347

PPO – Les hyperparamètres

- **Epsilon** le plus souvent 0.1 et 0.3 (valeur utilisée : 0.2)

Permet de ne pas faire de mise à jour trop drastique de notre politique

- **Lambda** (valeur utilisée : 0.95)

Sert dans le calcul de $\widehat{A}_t$ permet de ternir plus ou moins compte des récompenses potentielles futures (1 → tient compte de tout, 0→ ne tient compte que de la récompense à l'instant t)

- **Reward** (si la cible est touchée : +2)

- **Penalty** (valeur utilisée : -1 / max_step, avec max_step = 5000)

Sert à optimiser la distance parcourue sans trop pénaliser l'action

- **Batch** (valeur utilisée : 10)

Nombre d'expérience à collecter avant de mettre à jour le modèle

- **Learning rate** (valeur utilisée : 0.0003)

Pour mettre à jour les coefficients à la fin d'un batch

- **Beta** (valeur utilisée : 0.005)

Coefficient de l'entropie

Curiosity

- Récompenses assez clairsemée (très peu de retour pour que l'agent apprenne)
- Idée : Augmenter les récompenses extrinsèques avec signaux additionnels (plutôt dense) qui sont liés aux problématiques que l'agent doit résoudre

- Plusieurs stratégies ont été développée, dans ML-Agents il s'agit de la « **curiosity driven exploration** »
- L'idée est d'inciter l'agent à apprendre de nouvelle chose :
  - **ϵ-greedy exploration** (probabilité de tester)

  Commence à 1 et diminue avec le temps → probabilité(action aléatoire)= ϵ

  ϵ proche de 0, l'agent suit complètement la politique

  MAIS pas suffisant pour tout explorer si l'environnement est vaste/complexe

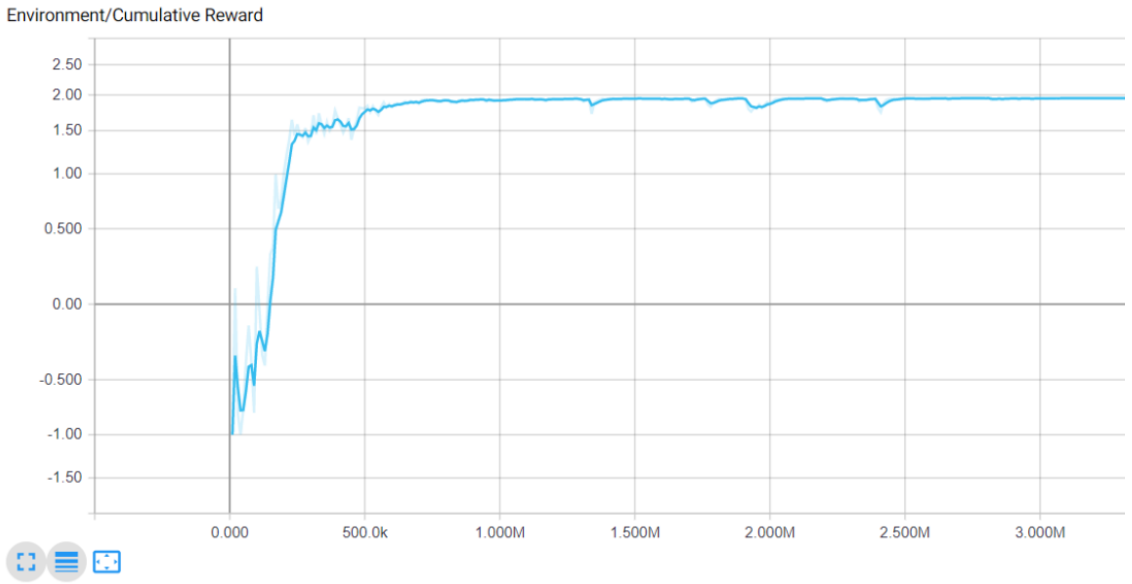
  - **Forward model** : l'agent voit un input frame spécifique et réalise une représentation latente de ce qu'il voit et dans le même temps *forward model* essaye de prédire cette représentation latente

  Si l'agent a déjà beaucoup exploré cet endroit, les prédictions du *forward model* vont être très bonnes

  Nouvel situation → prédiction moins bonne et possibilité d'utiliser ces erreurs en plus des récompenses pour inciter l'agent à explorer d'avantage (ajout de la surprise de l'agent vis-à-vis de ce qu'il s'est passé)

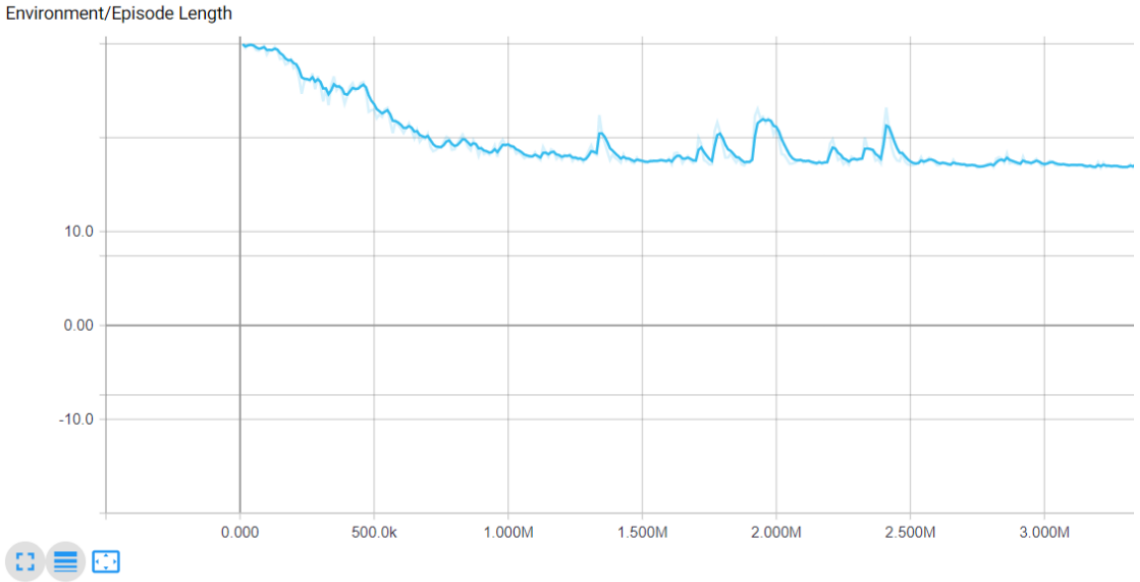
Source : arXiv:1705.05363

Tensor Board (1)



Evolution de la récompense obtenue

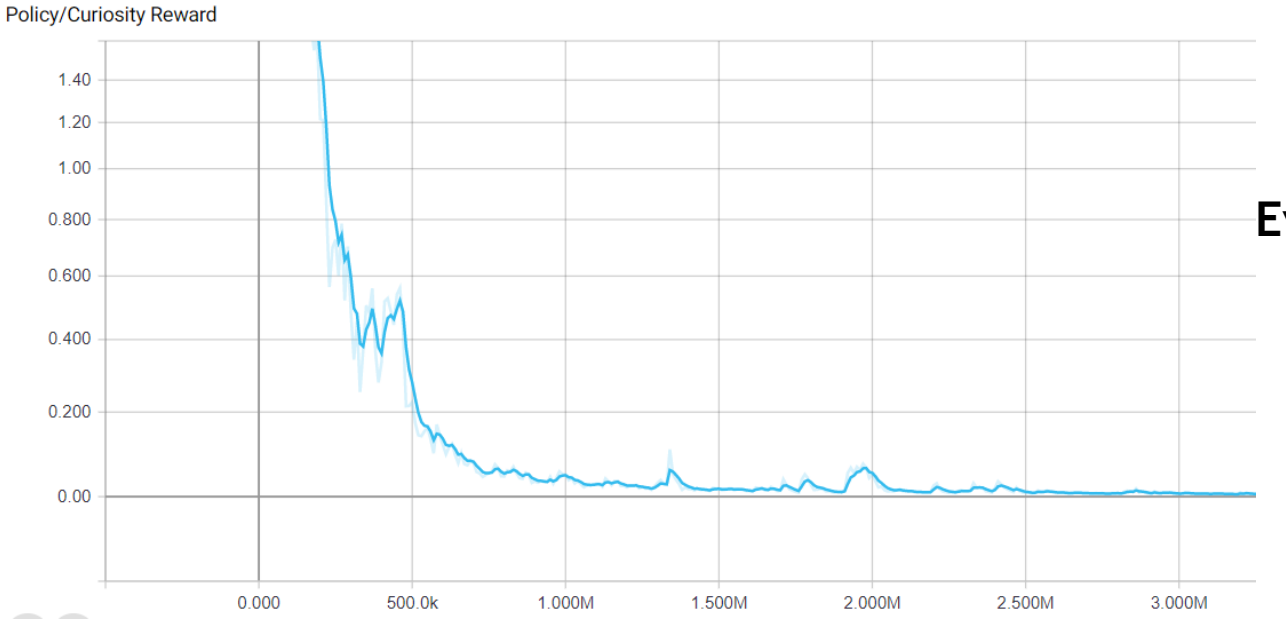
Valeur se rapproche de 2
(+2 si touche avec la cible mais -0.0002/action)



Evolution de la durée d’un jeu

Durée en temps (en sec)
Partie de plus en plus rapide puis plateau

Tensor Board (2)



Evolution de la curiosity

Evolution de la standard déviation par rapport à la récompense totale

```
INFO:mlagents.trainers:Saved Model
INFO:mlagents.trainers: racing2_cpu: CubeBrain: Step: 3350000. Time Elapsed: 28357.500 s Mean Reward: 1.950. Std of Reward: 0.014. Training.
INFO:mlagents.trainers: racing2_cpu: CubeBrain: Step: 3360000. Time Elapsed: 28440.759 s Mean Reward: 1.952. Std of Reward: 0.010. Training.
INFO:mlagents.trainers: racing2_cpu: CubeBrain: Step: 3370000. Time Elapsed: 28522.621 s Mean Reward: 1.951. Std of Reward: 0.012. Training.
```