IAR: Résumé d'article

Gym fourni un environnement de jeux Atari dans lequel l'agent doit apprendre à maximiser ses récompenses simplement en jouant le jeu. Les entrées que l'agent prend sont les pixels et le signal de récompense. Il a été montré récemment que des algorithmes NES peuvent donner des résultats qui concurrencent des algorithmes "classique" de RL. Les auteurs avaient pour but de montrer que même la classe d'algorithme beaucoup plus basique comme CES pouvait donner des résultats aussi satisfaisant.

-OpenAI's Natural Evolution Strategies: Cet algorithme de haut niveau fait évoluer une distribution de réseaux de politique en évaluant une population de $\boldsymbol{\lambda}$ réseaux différents à chaque itération. En effet à chaque itération il évalue le score de $\boldsymbol{\lambda}$ vecteurs de parametres de la politique centré autour d un paramètre initial θ. Aux scores est ensuite attribué un rang avec lequel une approximation de gradient va pouvoir être faite. La dernière étape de l'itération est la mise à jour du réseau de politiques.

-Canonical ES: Cet Algorithme très basique fait partie de la famille ($\boldsymbol{\mu}$,$\boldsymbol{\lambda}$) ES. On commence par initialiser un vecteur de paramètres $\boldsymbol{\theta}$, à chaque itération, on génère $\boldsymbol{\lambda}$ "clones"(descendant) auxquels on applique un bruit de "mutations" et sélectionne les $\boldsymbol{\mu}$ meilleurs. On simule ainsi le processus de sélection naturelle.

Les auteurs ont évalué les algorithmes sur 8 jeux Atari : Alien, BeamRider, Breakout, Enduro, Pong, Qbert, Seaquest et SpaceInvaders. Tous les jeux ont été évalués selon les mêmes protocoles qu'on peut résumer ainsi :

- Réutilisation du réseau de neurones de l'algorithme DQN, mais en modifiant la fonction d'activation en Exponential Linear Unit (ELU) et en ajoutant des couches de batch normalization. On se retrouve avec un réseau de 1.7 millions de paramètres et dont les poids sont initialisés avec une distribution normale d'espérance 0 et d'écart type 0.05
- Concernant le training, pour chaque jeu et chaque algorithme, on lance 3 runs, chacun sur 400 CPU et avec un budget de 10 heures
- Bien que la taille de la population parent μ soit un hyper paramètre important, on fixe μ à 50

On lance alors le training sur chaque jeu, on obtient 6 politiques par jeu (3 de OpenAI ES et 3 de Canonical ES) qui seront toutes évaluées sur 30 parties, on calcule ensuite les moyennes des scores :

- Pour Alien, Canonical ES est meilleur, les 3 politiques de Canonical ES sont en effet meilleures que les 3 d'OpenAI ES. (On dira que Canonical ES l'emporte avec 3 victoires contre 0)

- Pour BeamRider, c'est OpenAI ES qui l'emporte avec 3 victoires sur Canonical ES
- Pour Breakout, Canonical ES l'emporte avec 2 victoires contre 1
- Pour Enduro, OpenAI ES l'emporte avec 2 victoires contre 1
- Pour Pong, on a 2 victoires pour OpenAI ES et 1 ex aequo
- Pour Qbert, Canonical ES l'emporte avec 3 victoires contre 0
- Pour Seaquest, Canonical ES l'emporte avec 2 victoires contre 1
- Pour SpaceInvaders, OpenAI ES l'emporte avec 3 victoires contre 0

Pour notre projet nous utiliserons google cloud et nous nous demanderons si on peut obtenir les mêmes résultats ou similaire pour le jeu QBert, mais avec moins de budget.