Pacman prostředí pro metody strojového učení

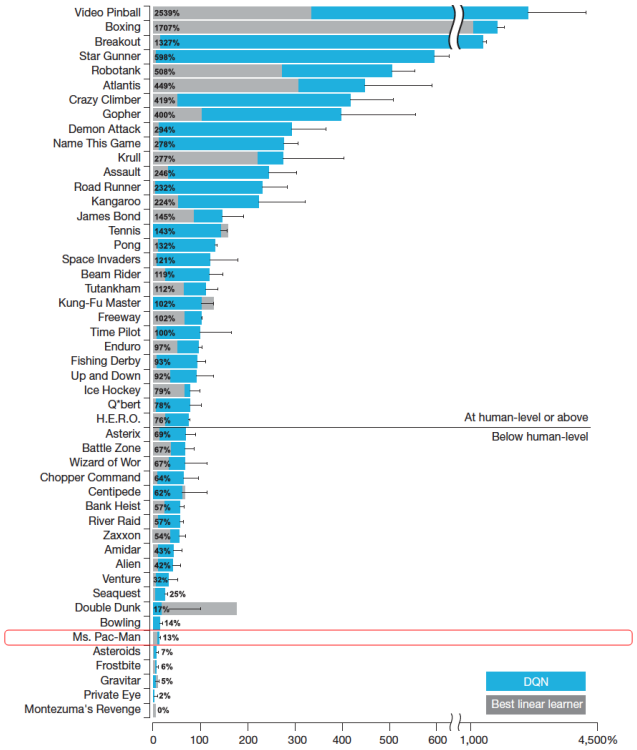
Filip Chodura
ČVUT–FIT
chodufil@fit.cvut.cz

7. ledna 2024

Mým úkolem bylo implementovat klasickou hru PacMan, tak aby se dala použít při trénování algoritmů posilovaného učení. Zároveň jsem implementoval algortimus DQN a ověřil jeho efektivitu na tomto prostředí.

1 Úvod

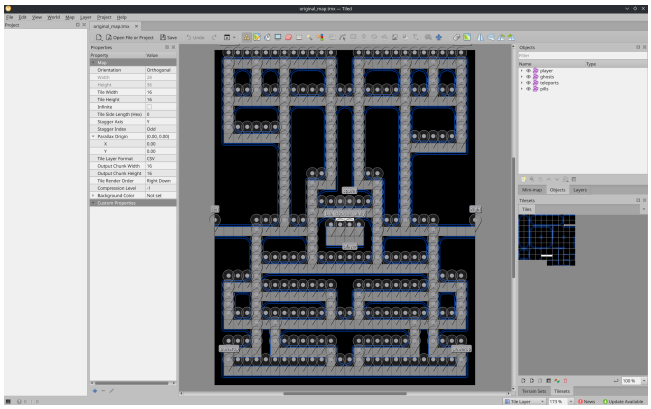
V dnešní době, se standartně pro trénování modelů na hře pacman používá implementace původní hry pro atari MsPacman, která je k dostání v knihovně gymnasium (dříve gym) od OpenAi. Problémem je, že tato implementace pouze emuluje původní atari a jako výstup se používá celá obrazovka, tedy pole trojic (RGB). Tyto modely se často trénují desítky hodin, dokonce i několik dní. Pacman je pro metody posilovaného učení složitá hra, jak ukazuje obrázek č. 1. Pravidla hry jsou pro AI komplikovaná. Jeden ze způsobu jak urychlit učení u těchto složitějších scénářů je na rozklad na jednodušší podsituace. U pacmana by to bylo například nechat model se učit první bez nepřátel,a poté je přidat atd. To tato implementace ale nepodporuje, proto jsem se rozhodl vytvořit svou vlastní implemenataci. Má implementace podporuje libovolný počet nepřátelé, počítání životů, zvuky, powerupy a teleporty z okrajů map.



Obrázek 1: Srovnání efektivity posilovaného učení na jednotlivých hrách [5]

2 Hra Pacman

Pacman je arkádová hra, která byla vydána roku 1980 v Japonsku. Úkolem hráče je posbírat všechny body na obrazovce a zároveň se vyhnout čtveřici duchů. Rozhodl jsem se implementovat Pacmana pomocí knihovny pygame jako modul v pythonu a zároveň dovolit uživateli si definovat vlastní úrovně pomocí editoru Tiled. Tiled je populární editor pro tvorbu videoherních úrovní. Tiled pracuje na principu vytváření a editace 2D tilemap, kde uživatel umisťuje malé obrázky (tiles) do vrstev, tvořících celkovou herní mapu. Dané mapy potom mohou být exportovány např. do formátu json (postup jak importovat mapu do mého pacmana jsem popsal v dokumentaci samotného modulu). Takto například vypadá mapa pacmana v Tiled 2. To nám umožňuje trénovat modely na jednoduchých mapách a až poté představit složitější scénáře.



Obrázek 2: Editor Tiled a tvorba úrovně

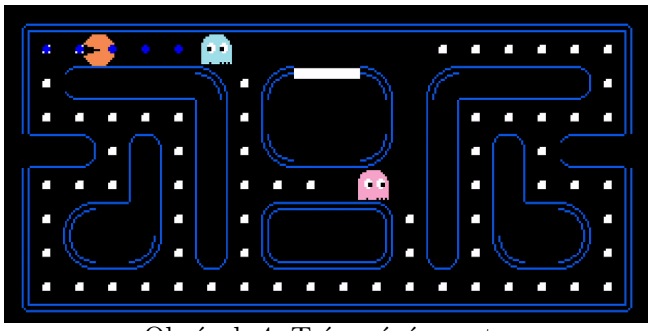
3 Nepřátelé

Tradiční nepřátelé pacmana jsou duchové Pinky, Inky, Clyde, Blinky. Pro jejich umělou inteligenci jsem využil dvou přístupů. Pro Pinkyho jsem implementoval heuristickou AI, která se podobá implementaci původní hry. V každém okamžiku se Pinky podívá, jaký tah může provést a vybere ten, který ho dostane nejblíže (euklidovská vzdálenost) k hráči. U ostatních jsem využil A* algoritmu pro hledání optimální cesty. Začíná s počátečním bodem a systematicky prochází možné cesty, vybírá ty, které mají nejnižší součet skutečné vzdálenosti a heuristiky (v mém případě euklidovská vzdálenost). Tím se zaměřuje na cesty, které jsou pravděpodobně nejkratší. Tento algoritmus jsem upravil,že pro zadaný cíl, ke kterému neexistuje žádna cesta je použitý dostatečně blízký bod. Tuto úpravu jsem provedl, aby nepřátelé mohli cílit například několik polí před nebo za hráče. Tím, že Inky cílí několik polí před hráče a Clyde za hráče, vytváří situace, kde se snaží oba hráče sevřít uprostřed, jak je vidět na obrázku č. 4. Při dostatečné blízkosti hráče se zaměří přímo na něj. Když pacman narazí na power-up, tak se přepnou duchové do scared mode. V mé implementaci se snaží dostat na pole, které je nejvzdálenější od hráče.

Obrázek 3: Umělá inteligence v Pacmanu. Barevné tečky představují výsledek A* algoritmu

4 Gymnasium Enviroment

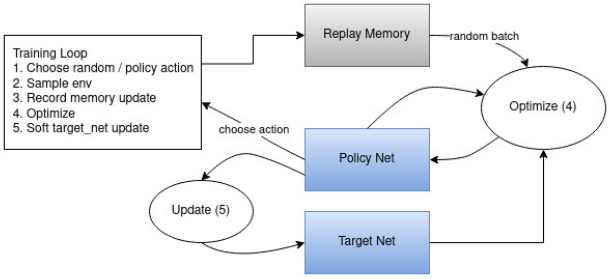
Jeden z nejčastěji využívaných přístupů k trénování modelů pro zesilované učení je pomocí knihovny Gymnasium (dříve známé jako Gym). Tato knihovna standartizuje rozhraní prostředí, na kterých mohou být spouštěny algoritmy pro zesilované učení. Každé prostředí poskytuje rozhraní pro interakci s agentem (algoritmem) a získávání zpětné vazby na jeho akce. Kromě rozsáhlé nabídky předdefinovaných prostředí umožňuje také snadné vytváření vlastních. Gym-pacman je tedy wrapper kolem mé implementace pacmana. Výstup prostředí je zjednodušené herní pole, kde poloha entit na mapě je zredukovaná na danou síť a je jim přiřazena hodnota. Vstup (action) tohoto prostředí je integer, který definuje směr hráče. Pro trénování definuju metodu step, která simuluje krok ve hře a vrací award, odměnu za daný krok, a další stav hry. Právě odměňování modelu, je jeden z nejdůležitějších hyperparametrů. Proto, abych hráče donutil k pohybu, jsem zvolil, že za kadžý snímek, co hráč nezíská žádne skore, bude penalizován. Penalizován je také za smrt. Odměněn je tedy za sebrání bodu.


Obrázek 4: Trénování agenta

5 Deep Q Network DQN

Jako jeden z nejefektivnějších algortimů strojového učení na hrách je DQN. Tento algortimus jsem se snažil co nejjednodušeji implementovat (ukázalo se jako mnohem větší problém, než jsem předpokládal) pomocí knihovny pytorch. DQN používá hlubokou neuronovou síť k aproximaci Q-hodnoty pro každou akci v daném stavu. Q-hodnota je odhad budoucí odměny, kterou může agent očekávat, když provede konkrétní akci v daném stavu prostředí. Do Replay memory jsou ukládány vzorky (stav, akce, odměna, nový stav) z interakcí s prostředím. Z této paměti je vybrán náhodně batch. DQN využívá dvě hluboké sítě - policy net a target net. Policy net rozhoduje o volbě akce na základě aktuálního výstupu prostředí. Target Network má stejnou architekturu jako policy net, ale její parametry jsou aktualizovány mnohem

pomaleji, aktualizuje parametry policy net a stabilizuje učení. DQN používá epsilon-greedy strategii„ kde velikost paramateru epsilon ovlivňuje, jak moc se síť bude snažit zkoumat nové možnosti v prostředí (vybírat náhodné směry v případě pacmana) nebo se bude držet naučených vlasntostí. Celý cyklus je zobrazen na obrázku č. 5



Obrázek 5: Proces DQN [4]

6 Závěr

Implementoval jsem hru pacman-game, gymnasium prostředí gym-pacman a na něm jsem natrénoval algoritmus DQN. Experimentoval jsem s různými odměňovacími funkcemi pro pacmana, s espilon hyperparametrem, s pohybem samotného hráče. Minimalizoval jsem trénovací prostředí pro snadnější pochopení hry algoritmem. Bohužel jsem nedokázal natrénovat příliš efektivní model. To především usuzuji krátké době trénování. Rozhodně se v mém projektu nachází velký prostor pro lepší ladění hyperparametrů agenta a celkově by bylo výhodné rozšířit celé testovací rozhraní. Pro mě se jednalo pro první seznámení s reinforcment learningem a svůj výsledek považuji za úspěšný.

Reference

[1] Pygame sprite documentation. online. [cit. 2024-01-07] `https://www.pygame.org/docs/ref/sprite.html`.

[2] Tiled map editor. online. [cit. 2024-01-07] `https://www.mapeditor.org/`.

[3] Medium. How to train ms. pac-man with reinforcement learning. online. [cit. 2024-01-07] `https://medium.com/analytics-vidhya/how-to-train-ms-pacman-with-reinforcement-learning-dea714a2365e`.

[4] PyTorch. Pytorch reinforcement q-learning tutorial. online. [cit. 2024-01-07] `https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html`.

[5] Google Research. From pixels to actions: Human-level control through deep reinforcement learning. online, 2015. [cit. 2024-01-07] `https://blog.research.google/2015/02/from-pixels-to-actions-human-level.html?m=1`.