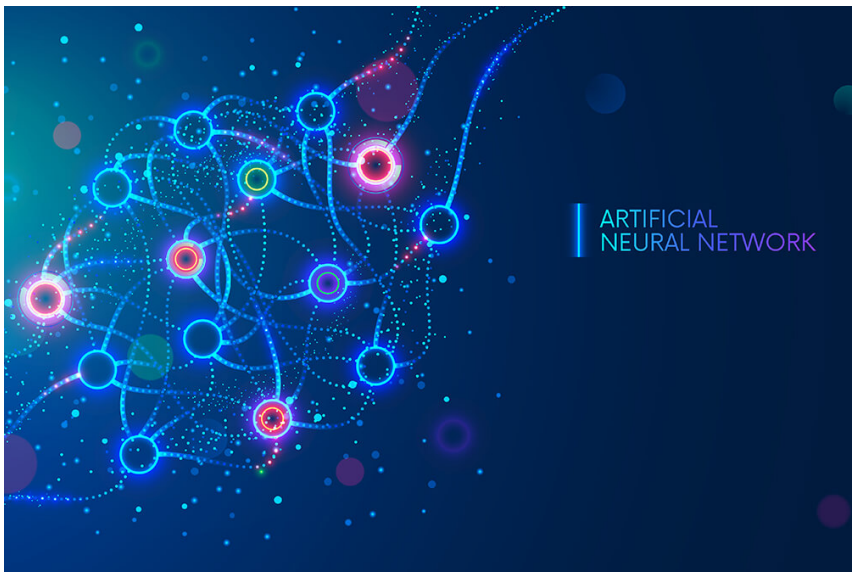
SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

NeuralPath Labyrinth



Autor: Lukáš Hrňa

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2023/24

Poděkování

Prostor k poděkování Tučňákovi.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 1. 1. 2024

. . . . . .

Podpis autora

Abstrakt

Tento projekt se zaměřuje na aplikaci umělé inteligence pro řešení jednoduchých bludišť. Využíváme PyTorch, knihovnu pro strojové učení, a 'Gym' od OpenAI, nástroj pro vývoj a testování algoritmů posilovaného učení. Cílem je vyvinout a trénovat model AI, který efektivně prochází prostředím generovaných bludišť. Model je navržen pro rozpoznávání vzorů a navigaci různými konfiguracemi bludišť s cílem najít cestu ven.

Klíčová slova

Umělá inteligence, Posilované učení, PyTorch, OpenAI Gym, Navigace bludištěm, Hluboké neuronové sítě, Prostorová navigace, Algoritmy pro hledání cesty, Generování bludišť, Rozpoznávání vzorů

Abstract

This project focuses on the application of artificial intelligence to solve simple mazes. We utilize PyTorch, a machine learning library, and 'Gym' from OpenAI, a tool for developing and testing reinforcement learning algorithms. The goal is to develop and train an AI model capable of efficiently navigating through the environment of generated mazes. The model is designed to recognize patterns and navigate through various maze configurations to find a way out.

Keywords

Artificial Intelligence, Reinforcement Learning, PyTorch, OpenAI Gym, Maze Navigation, Deep Neural Networks, Spatial Navigation, Pathfinding Algorithms, Maze Generation, Pattern Recognition

Obsah

Úvod

Závěrečná studijní práce na Střední škole průmyslové a umělecké v Opavě je významnou částí vzdělávacího procesu v oblasti informačních technologií. Tento projekt zkoumá aplikaci umělé inteligence v řešení jednoduchých bludišť s využitím nástrojů jako PyTorch a OpenAI Gym. Cílem je vyvinout AI model pro efektivní navigaci v bludišti, zaměřující se na základní implementaci a testování AI algoritmů bez použití inovativních nebo pokročilých metod. Tento přístup nabízí praktický pohled na využití AI v jednoduchých aplikacích, přinášející užitečné poznatky pro řešení specifických problémů.

První kapitola poskytuje teoretický základ, nezbytný pro pochopení použitých principů umělé inteligence a posilovaného učení v rámci projektu. Detailně zkoumá nástroje PyTorch a OpenAI Gym, vysvětluje jejich role a funkce v kontextu řešení bludišť. Tato kapitola také poskytuje základní teoretické informace o neuronových sítích, jejich struktuře a principů fungování. Zahrnuje přehled klíčových konceptů posilovaného učení, které jsou důležité pro následnou praktickou část.

Druhá kapitola je plně zaměřena na samotný projekt. Tato kapitola popisuje proces vývoje, implementaci a testování AI modelu, který byl navržen pro efektivní navigaci v bludišti. Podrobně představuje specifické výzvy a metody, které byly použity při vytváření a trénování modelu, a zkoumá jeho výkon a schopnost adaptace na různé typy bludišť. Závěr kapitoly se věnuje analýze výsledků a diskusi o zjištěních a možných směrech pro další vývoj.

1 UMĚLÉ NEURONOVÉ SÍTĚ

1.1 ÚVOD

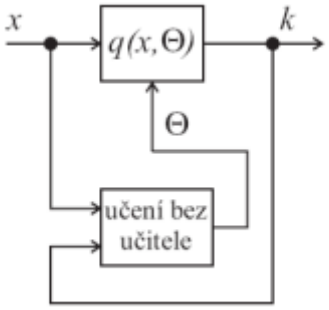
Tato kapitola slouží jako úvod do tématu umělých neuronových sítí. Zabývá se vysvětlením základních pojmů, popisem matematického modelu neuronu a jeho schopností klasifikace. Klíčovým prvkem je pochopení matematického modelu jednoho neuronu, což je základ pro pochopení komplexnějších struktur tvořených propojenými neurony v rámci umělých neuronových sítí

1.2 TYPY UČENÍ

Učení v umělé inteligenci lze rozdělit do několika základních kategorií, z nichž nejvýznamnější jsou učení s učitelem, bez učitele a posilované učení. Každý typ má své specifické charakteristiky a využití.

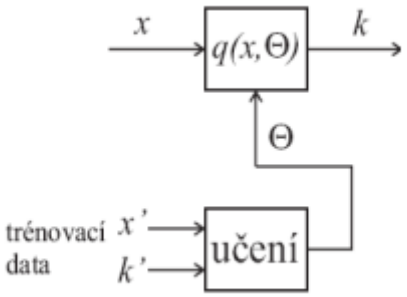
1.2.1 Unsupervised learning

Učení bez učitele, nebo unsupervised learning, pracuje s neoznačenými daty. Cílem je najít skryté vzory nebo struktury v datech, aniž by byly k dispozici předem definované odpovědi nebo labely. Typickými příklady využití jsou klastrování, kde je úkolem najít skupiny podobných prvků v datech, nebo redukce dimenzionality, která slouží k zjednodušení komplexních datových sad. Učení bez učitele je klíčové v situacích, kdy je třeba rozpoznat inherentní strukturu dat bez předchozích znalostí o tom, co data přesně obsahují.



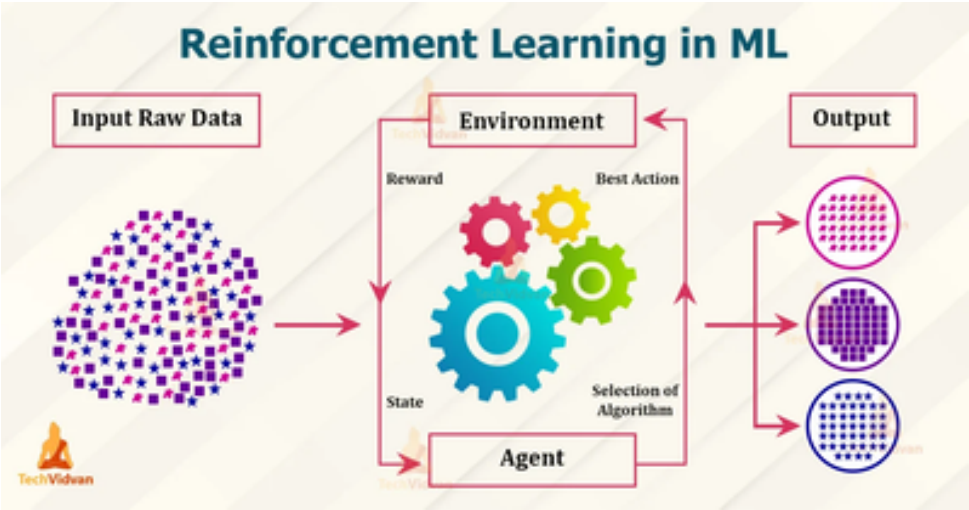
1.2.2 Supervised learning

Učení s učitelem, nebo supervised learning, je proces, kde model umělé inteligence je trénován na datové sadě, která obsahuje jak vstupní data, tak správné odpovědi (labely). Tento přístup umožňuje modelu naučit se předpovídat výstupy na základě nových vstupů na základě předchozích zkušeností. Tento typ učení je široce využíván v aplikacích jako je klasifikace (například rozpoznávání objektů na obrázcích) a regrese (například předpovídání cen nemovitostí), kde je důležité, aby model byl schopen správně identifikovat a reagovat na různé druhy dat.



1.2.3 Reinforcement learning

Posilované učení, nebo reinforcement learning, je zase odlišné tím, že se zaměřuje na vývoj algoritmů, které se učí prostřednictvím interakce s prostředím a získávají zpětnou vazbu v podobě odměn nebo trestů. V tomto přístupu je agent, kterým může být například robot nebo software, vystaven prostředí, ve kterém se snaží provádět akce vedoucí k maximální možné odměně. Posilované učení je ideální pro situace, kde jsou rozhodnutí časově závislá a kde je třeba vzít v úvahu dlouhodobé důsledky akcí, jako jsou strategické hry, robotická navigace nebo autonomní řízení vozidel.



Pro maximalizaci celkové odměny je nezbytné, aby agent volil optimální akce v závislosti na svém současném stavu. Tento výběr akcí, známý jako strategie nebo politika π , je klíčový pro přiřazení nejvhodnější akce pro každý stav. Existuje mnoho algoritmů pro určení efektivní funkce π , nicméně zde se zaměříme specificky na Q-learning a jeho odvozené algoritmy.

1.3 Q-LEARNING

Q-learning je algoritmus založený na principu posilovaného učení. Na rozdíl od metod, které přímo hledají optimální strategii π (metody policy gradient), Q-learning postupuje nepřímo prostřednictvím tzv. Q-tabulky. Tato tabulka spojuje stavy s akcemi - každý řádek reprezentuje jeden stav a sloupce v tomto řádku různé akce. Agent vybírá akci s nejvyšší hodnotou v tabulce. Tyto hodnoty jsou známé jako Q-hodnoty, kde "Q"reprezentuje "quality"(kvalitu) akce. Algoritmus, který určuje nejlepší akci z Q-tabulky, je označován jako Q-funkce. Aktualizace

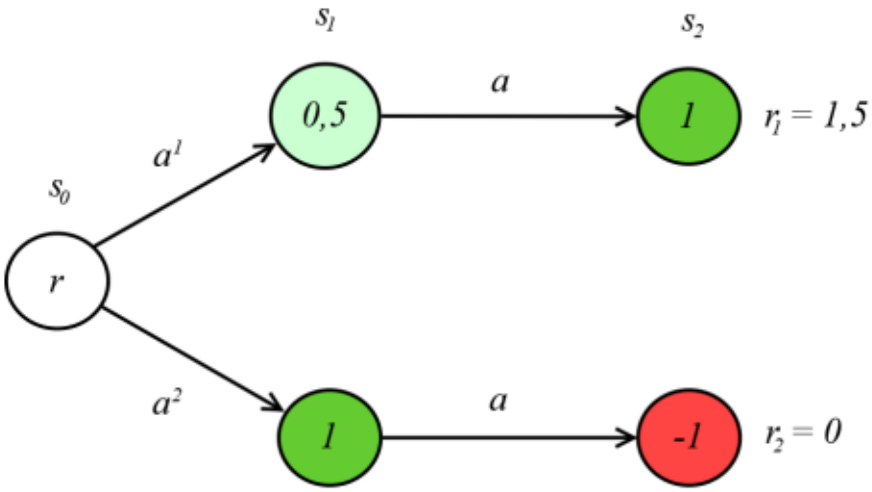
States	Actions			
	Left	Right	Up	Down
Left	0	0	0	1
Right	0	0	1	0
Up	-1	0	1	0
Down	-1	0	0	0
Food Left	2	0	0	0
Food Right	0	5	0	0
Food Up	1	0	0	0
Food Down	0	0	0	0
...				

Q-hodnot se provádí na základě odměn, které agent obdrží od prostředí za provedenou akci. Tato odměna, známá jako posílení, může být pozitivní, negativní (trest) nebo neutrální. Když agent obdrží za určitou akci negativní odměnu, dojde k snížení příslušné hodnoty v Q-tabulce, což vede k nižší pravděpodobnosti opětovného výběru této akce. Takto lze postupně eliminovat nežádoucí chování a podporovat chování, které je žádoucí.

1.3.1 Okamžitá a dlouhodobá odměna

Aktualizace Q-hodnot lze rozdělit na dvě zásadní části: okamžitou a dlouhodobou. Okamžitá část je reprezentována odměnou r_t, kterou agent obdrží za provedení akce a_t. K této okamžité

odměně je pak přičítána dlouhodobá hodnota, reprezentující očekávané budoucí Q-hodnoty ze stavu s_{t+1}. Cílem je maximalizovat celkovou odměnu, zahrnující nejen aktuální, ale i budoucí akce. Například, v případě dvou možných akcí (a_1, a_2) ve stavu s_0 může být na první pohled akce s vyšší okamžitou odměnou méně výhodná v dlouhodobém horizontu, pokud by v důsledku jejího provedení následovaly stavy vedoucí k nižším celkovým odměnám.



[h]

Je klíčové najít správnou rovnováhu mezi okamžitou a dlouhodobou odměnou. Úloha dlouhodobé odměny může být upravena pomocí diskontního faktoru (discount factor) γ, který určuje váhu budoucích odměn, zatímco míra učení (learning rate) α ovlivňuje rychlost aktualizace Q-hodnot. Celkovou odměnu r, skládající se z okamžité a dlouhodobé složky, lze vyjádřit jako součet těchto dvou komponent, s přihlédnutím k váze diskontního faktoru a míry učení.

$$r = a.(r_t + \gamma.maxQ(s_{t+1,a})) \tag{1.1}$$

V Q-tabulce se Q-hodnota pro daný stav a akci obvykle nepřepisuje přímo novou odměnou, ale upravuje se na základě již existující hodnoty. Toto se provádí za účelem zachování historických informací o výkonnosti dané akce. Aktualizační pravidlo pro Q-hodnoty tedy bude kombinovat aktuální odměnu s předešlými poznatky o daném stavu a akci, čímž se dosahuje efektivnějšího a přesnějšího učení.

$$Q^{nova}(s_t, a_t) = (1-\alpha).Q(s_t, at) + r \tag{1.2}$$

1.3.2 Limitace Q-learningu

Při použití Q-tabulky pro mapování stavů na akce v rámci Q-learningu se můžeme setkat s několika komplikacemi. Pro složitější úlohy, kde je stav určen několika proměnnými současně (například v 3D prostoru), může velikost Q-tabulky dosáhnout extrémních rozměrů, což vede k její nepraktičnosti. Navíc, protože stavy musí tvořit konečnou sadu, řešení spojitých stavů se stává problematickým. Další výzvou je manipulace s akcemi: v základním modelu Q-learningu jsou akce pouze přijaty nebo odmítnuty (binární výběr), což znemožňuje spojité řízení výstupu. Tyto problémy mohou být řešeny implementací umělých neuronových sítí, které nabízí flexibilnější a efektivnější přístup.

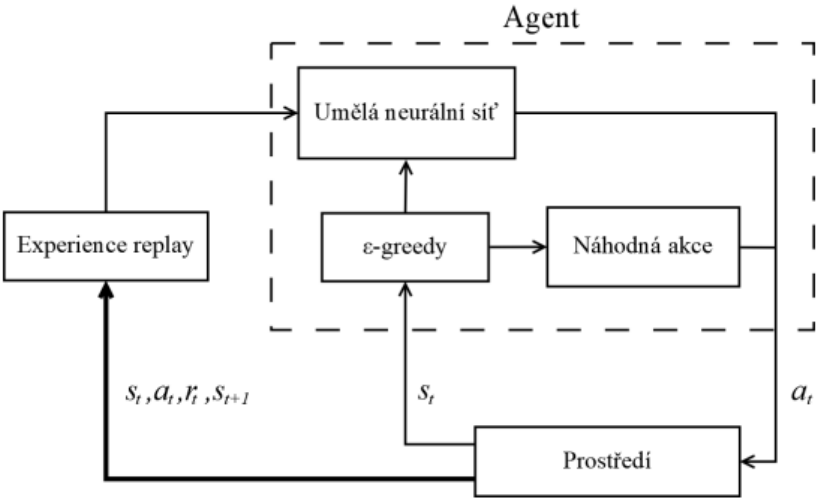
1.3.3 Deep Q-network

Deep Q-network (DQN) je algoritmus, který byl vytvořen společností DeepMind v roce 2015. Tento algoritmus spojuje principy Q-learningu s umělou neuronovou sítí. Základním konceptem DQN je využití neuronové sítě, známé jako Q-síť, která nahrazuje tradiční Q-tabulku. Úkolem této sítě je co nejvíce přiblížit funkci Q-hodnoty. Vstupem do Q-sítě je stav s_t, a výstupem je reakce na tento stav a_t, což je podobné Q-tabulce. Nicméně, proces učení Q-sítě se liší od Q-tabulky, protože zde nelze jednoduše aktualizovat Q-hodnoty. Místo toho se Q-síť musí učit stejným způsobem jako jakákoliv jiná neuronová síť, a to využitím učení s učitelem, kde jsou všechny akce agenta stále hodnoceny pomocí odměňovací funkce a generují trénovací data.

Další rozdíl oproti klasickému Q-learningu je v aktualizačním pravidle. Vzhledem k tomu, že neuronovou síť neměníme přímo, vynecháváme část aktualizačního pravidla, která připočítává starou Q-hodnotu. Toto pravidlo aktualizace pak nabývá nové formy, která efektivně využívá schopnosti neuronové sítě k učení a přizpůsobení.

$$Q^{nova}(s_t, a_t) = r_t + \gamma max Q(s_{t+1}, a) \tag{1.3}$$

Neuronové sítě jsou efektivnější, když se učí na základě více dat, než jen jednoho výstupu. Efektivnějším přístupem je uchovávání historie předchozích stavů agenta. Zaznamenáváme stav, akci, odměnu a následující stav (s_t, a_t, r_t, s_{t+1}) - všechny elementy potřebné pro výpočet nových Q-hodnot. Během učení pak agent vybírá náhodně uspořádanou vzorku pevně stanovené velikosti z těchto uložených záznamů. Na základě této vzorky se počítají nové Q-hodnoty, které jsou poté uplatněny na neuronovou síť pomocí algoritmu zpětného šíření chyb. Tento proces uchovávání minulých stavů se nazývá metoda opakování zkušeností (experience replay).



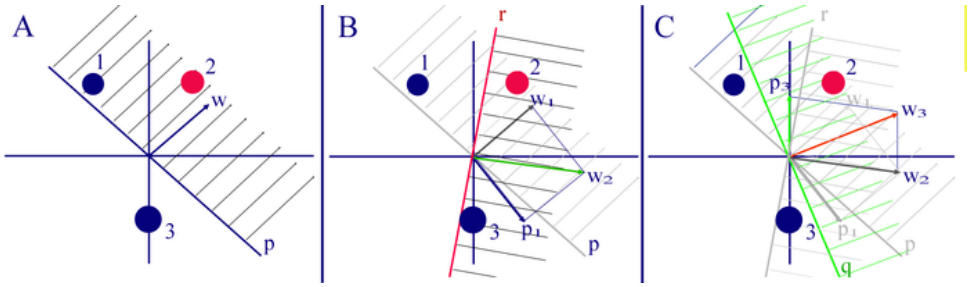
1.4 PERCEPTRON

Perceptron v nejjednodušší podobě je binární klasifikátor, který mapuje vektory vstupů $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ na výstupní hodnoty $f(\xi)$:

$$f(\xi) = \begin{cases} 1 & \text{pro } \mathbf{w} \cdot \mathbf{x} - \theta \geq 0 \\ 0 & \text{jindy} \end{cases}$$

kde $\mathbf{w}$ je vektor vah a θ je práh citlivosti neuronu.

Celkový podnět neuronu udává vážený součet $\xi = \sum_{i=1}^{n} w_i \cdot x_i - \theta$. Tento celkový podnět bývá označován jako potenciál neuronu. Na potenciál reaguje neuron (perceptron) výstupní odezvou $Z = f(\xi)$, kde f je tzv. přenosová funkce. Je-li přenosová funkce ve tvaru skokové funkce, perceptron funguje, jak již bylo řečeno, jako binární klasifikátor, neuron tedy dělí vstupní prostor na dvě části. Tři fáze učení jednoduchého perceptronu jsou následující:



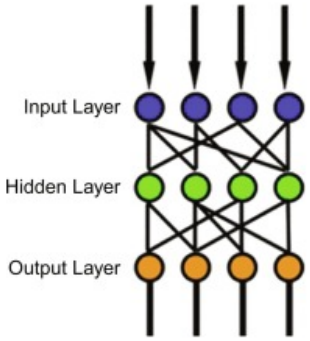
A: Je zvolen náhodně vektor vah $\mathbf{w}$ a k němu určena kolmá rozhodovací hranice p. Zjišťujeme výstup pro bod 1 - leží v oblasti s výstupem 1, i když má ležet v oblasti s výstupem 0.

B: Odečteme vektor bodu 1 od vektoru vah $\mathbf{w_1}$ a získáme nový vektor vah $\mathbf{w_2}$ a k němu příslušnou rozhodovací hranici znázorněnou přímkou r. Bod 2 je umístěn správně, bodu 3 přiřadí síť hodnotu 1 i když má dostat výstup 0.

C: Odečteme vektor bodu 3 od vektoru vah $\mathbf{w_2}$ a získáme nový vektor vah $\mathbf{w_3}$. Nyní je již problém vyřešen - všem bodům je přiřazen odpovídající hodnota výstupu. Řešením problému je tedy vektor vah $\mathbf{w_3}$ s příslušnou rozhodovací hranicí q.

1.5 MULTI LAYER PERCEPTRON(MLP)

MLP je rozšířením dopředné neuronové sítě. Skládá se ze tří typů vrstev - vstupní vrstvy, výstupní vrstvy a skrytých vrstev. Vstupní vrstva přijímá vstupní signál k zpracování. Výstupní vrstva provádí požadované úlohy, jako jsou predikce a klasifikace. Libovolný počet skrytých vrstev, umístěných mezi vstupní a výstupní vrstvou, představuje skutečný výpočetní motor MLP. Podobně jako u dopředné sítě v MLP proudí data směrem dopředu od vstupní k výstupní vrstvě. Neurony v MLP jsou trénovány pomocí algoritmu učení zpětné propagace. MLP jsou navrženy tak, aby aproximovaly jakoukoli spojitou funkci a mohly řešit problémy, které nejsou lineárně separovatelné. Hlavními využitími MLP jsou klasifikace vzorů, rozpoznávání, predikce a aproximace.



$$o_x = G(b_2 + W_2 h_x) \tag{1.4}$$

$$h_x = \Phi(x) = s(b_1 + W_1 x) \tag{1.5}$$

s vektory zkreslení b_1, b_2; maticemi vah W_1, W_2 a aktivačními funkcemi G a s. Sada parametrů k učení je sada $\theta = \{W_1, b_1, W_2, b_2\}$. Typické volby pro s zahrnují tanh funkci s $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$ nebo logistickou sigmoidní funkci, s $\text{sigmoid}(a) = \frac{1}{1+e^{-a}}$.

2 VYUŽITÉ POSTUPY A TECHNOLOGIE

V průběhu projektu byly využity následující postupy a technologie:

- **Umělé neuronové sítě:** Využití různých typů neuronových sítí pro modelování složitých vztahů a rozpoznávání vzorů.
- **Posilované učení:** Implementace algoritmů posilovaného učení, jako je Q-learning, pro efektivní rozhodování a adaptaci modelu.
- **Python a knihovny pro strojové učení:** Použití programovacího jazyka Python spolu s knihovnami jako PyTorch a Gym od OpenAI pro vývoj a trénování modelů.
- **Simulační prostředí:** Využití simulací pro testování a optimalizaci algoritmů umělé inteligence.
- **Data Analytics:** Analýza a zpracování dat pro lepší porozumění problému a efektivnější učení modelů.

2.1 2D BLUDIŠTĚ OVLÁDANÉ UMĚLOU INTELIGENCÍ

2.1.1 Vytvoření a interakce s prostředím

V rámci tohoto projektu je implementováno 2D bludiště, jehož navigaci řídí umělá inteligence. Pro vývoj a trénink AI modelu je využita knihovna PyTorch, což je oblíbený nástroj pro strojové učení napsaný v Pythonu, poskytující rozsáhlé možnosti pro práci s neurálními sítěmi.

Pro simulaci interakce AI s bludištěm využíváme knihovnu Gym od OpenAI, která poskytuje standardizované rozhraní pro různé druhy simulovaných prostředí. Gym je rovněž napsána v Pythonu a je navržena tak, aby usnadňovala vývoj a testování algoritmů posilovaného učení.

2.1.2 Ovládání bludiště

Interakce mezi AI a bludištěm je realizována pomocí knihovny pynput. Tato knihovna umožňuje AI simulovat stisknutí kláves, čímž se umělá inteligence "naučí"ovládat bludiště. Pynput je psána v Pythonu a poskytuje jednoduchý způsob, jak generovat klávesové události, což je klíčové pro integraci AI s prostředím bludiště.

2.1.3 Vykreslení prostředí

Pro vykreslení bludiště byly použity dva různé herní enginy – Raylib a Pygame. Raylib je C knihovna, která se zaměřuje na snadné a rychlé vytváření aplikací a her, zatímco Pygame je sada Python modulů určených pro tvorbu her. Obě tyto knihovny byly vybrány pro jejich efektivitu a schopnost plynule vykreslovat grafické prvky bludiště, což je nezbytné pro realistickou simulaci a interakci AI s prostředím.

ZÁVĚR

V rámci této práce jsme se zabývali využitím umělých neuronových sítí pro navigaci v bludišti, přičemž jsme se zaměřili na základní teoretické aspekty a praktickou aplikaci těchto technologií. Demonstrace AI modelu pro řešení bludišť zdůraznila potenciál neuronových sítí a současně odhalila výzvy spojené s jejich použitím.

LITERATURA

[1] DOKULIL Jakub. *Šablona pro psaní SOČ v programu LaTeX* [Online]. Brno, 2020 [cit. 2020-08-24]. Dostupné z: https://github.com/Kubiczek36/SOC_sablona

[2] OETIKER, Tobias, Hubert PARTL, Irene HYNA, Elisabeth SCHEGL, Michal KOČER a Pavel SÝKORA. *Ne příliš stručný úvod do systému LaTeX2e* [online]. 1998 [cit. 2020-08-24]. Dostupné z: https://www.jaroska.cz/elearning/informatika/typografie/lshort2e-cz.pdf

[3] *Wikibooks: LaTeX* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-08-24]. Dostupné z: https://en.wikibooks.org/wiki/LaTeX

[4] *TeX - LaTeX Stack Exchange* [online]. Stack Exchange, 2020 [cit. 2020-09-01]. Dostupné z: https://tex.stackexchange.com

[5] *Střední škola průmyslová a umělecká Opava* [online]. [cit. 2023-11-11]. Dostupné z: https://www.sspu-opava.cz

[6] *Citace PRO* [online]. Citace.com, 2020 [cit. 2020-08-31]. Dostupné z: https://www.citacepro.com

[7] BORN, Max a Emil WOLF. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. 7th (expanded) edition. Reprinted wirth corrections 2002. 15th printing 2019. Cambridge: Cambridge University Press, 2019. ISBN 978-0-521-64222-4.

Seznam obrázků

Seznam tabulek

PŘÍLOHA A SPOT DIAGRAMY A DALŠÍ