DELTA – Střední škola informatiky a ekonomie, s.r.o.

Ke Kamenci 151, Pardubice

Vizualizace fungování jednoduché neuronové sítě při řízení hry pong

Příjmení, jméno	**Jarešová Gabriela**
Třída	**4.A**
Studijní obor	**Informační technologie 18-20-M/01**
Školní rok	**2025/2026**

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení: *Gabriela Jarešová*

Pro školní rok: *2025/2026*

Třída: *4. A*

Obor: *Informační technologie 18-20-M/01*

Téma práce: *Vizualizace fungování jednoduché neuronové sítě při řízení hry*

Vedoucí práce: *RNDr. Jan Koupil, Ph.D.*

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cíl projektu: Cílem projektu je navrhnout a vytvořit aplikaci, která ukáže, jak se jednoduchá neuronová síť učí a ovládá chování agenta ve zvolené jednoduché počítačové hře. Zároveň bude umožněno sledovat, co se „v hlavě“ sítě děje – bude tedy vytvořen nástroj pro vizualizaci průběhu aktivací jednotlivých neuronů. Aplikace bude sloužit jako výukový nástroj pro základní seznámení s fungováním neuronových sítí a reinforcement learningu.

Specifikace projektu:

1. **Výběr herního prostředí:**
   - Bude vybrána jednoduchá hra s dobře definovaným stavovým prostorem (např. Pacman, FlatEthernet, Snake, Pong).
   - Bude provedena implementace nebo adaptace herního prostředí, aby bylo možné jednoduše číst herní stav a posílat akce agenta.
2. **Návrh a implementace neuronové sítě:**
   - Bude navržena jednoduchá neuronová síť (např. malá plně propojená síť), která bude zpracovávat stav hry a navrhovat akci.
   - Bude implementována základní forma učení (např. reinforcement learning – Q-learning s neuronovou aproximací nebo jednoduchá evoluční strategie).
   - Budou uchovávány informace o změnách vah v čase, případně statistikách učení.

3. **Vizualizace fungování sítě:**
    - Vizualizace fungování sítě:
        - Aktivaci jednotlivých neuronů ve vrstvě (např. barvou nebo jasem),
        - Změnu vah spojů (např. tloušťkou nebo barvou),
        - Průběh rozhodovacího procesu (např. zvýraznění vybrané akce).
    - Vizualizace bude propojena s běžící hrou v reálném čase.
4. **Režimy použití:**
    - Bude umožněno přepínat mezi ručním ovládáním hry (pro účely demonstrace a testování) a automatickým režimem, kdy hru ovládá neuronová síť.
    - Bude umožněno sledovat vývoj sítě během učení i přehrávat naučené chování.
5. **Výzkum a volba technologie:**
    - Bude proveden výzkum vhodných knihoven pro neuronové sítě a vizualizaci (např. TensorFlow.js, PyTorch, Keras, matplotlib, custom canvas).
6. **Testování a dokumentace:**
    - Bude provedeno testování chování sítě a vizualizačního nástroje na různých konfiguracích.
    - Bude vytvořena uživatelská a technická dokumentace, popisující implementaci, způsob učení i použití systému.
7. **Prezentace a obhajoba:**
    - Bude připravena demonstrace výsledného systému včetně ukázek naučeného chování.
    - V prezentaci bude vysvětlen návrh architektury a princip učení.

Hodnocení:

- Funkčnosti a srozumitelnosti vizualizace
- Kvality implementace hry a řízení agenta

- Srozumitelnosti a přínosnosti dokumentace
- Originality zpracování a přínosu pro výuku
- Kvality prezentace a schopnosti vysvětlit klíčové principy

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

- **Září:** Průzkum her vhodných pro řízení agentem, Výběr algoritmu a knihoven pro neuronovou síť
- **Říjen:** Implementace herního prostředí a jeho rozhraní, Návrh struktury sítě a způsobu řízení agenta
- **Listopad:** Implementace základní sítě a prvního učení, Vizualizace základních stavů (např. aktivace neuronů)
- **Prosinec - Leden:** Dopracování učení, práce s více vrstvami, přidání režimu přehrávání, Vylepšení vizualizací, přidání ovládacích prvků a popisků
- **Únor:** Testování, ladění, pokusy s různým nastavením sítě
- **Březen:** Tvorba dokumentace, demonstračních materiálů a prezentace

Prohlášení

Prohlašuji, že jsem svou práci vypracovala samostatně a použila jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

V Pardubicích dne _____ _____

Jarešová Gabriela

Poděkování

Ráda bych poděkovala svému vedoucímu projektu RNDr. Janu Koupilovi, Ph.D. za cenné rady, trpělivost a podporu při realizaci tohoto projektu.

Zvláštní poděkování také patří mé rodině za podporu během tohoto školního roku i celého studia.

Anotace

Cílem projektu je vytvoření výukové aplikace, která umožňuje sledovat v reálném čase, jak neuronová síť zpracovává informace a dělá rozhodnutí během hraní hry Pong. Aplikace kombinuje tři hlavní komponenty: herní engine implementovaný v Pythonu, agent využívající algoritmus Deep Q-Learning pro autonomní hraní, a vizualizační nástroj zobrazující aktivaci jednotlivých neuronů a sílu propojení mezi vrstvami sítě. Systém je určen především pro studenty a pedagogy jako názorný prostředek pro pochopení principů fungování neuronových sítí a reinforcement learningu.

Klíčová slova

Neuronové sítě, Deep Q-Learning, Reinforcement Learning, Vizualizace, PyTorch, Pygame, Python, Umělá inteligence, Strojové učení, Pong

Abstract

The aim of the project is to create an educational application that allows users to observe in real time how a neural network processes information and makes decisions while playing the game Pong. The application combines three main components: a game engine implemented in Python, an agent using the Deep Q-Learning algorithm for autonomous gameplay, and a visualization tool displaying the activation of individual neurons and the strength of connections between network layers. The system is primarily intended for students and educators as an illustrative tool for understanding the principles of neural networks and reinforcement learning.

Keywords

Neural networks, Deep Q-Learning, Reinforcement Learning, Visualization, PyTorch, Pygame, Python, Artificial Intelligence, Machine Learning, Pong

Obsah

1 Úvod

Cílem této práce je vytvořit interaktivní výukový nástroj, který umožní sledovat fungování neuronové sítě v reálném čase na příkladu jednoduché počítačové hry Pong. Aplikace kombinuje tři hlavní komponenty: herní prostředí, kde neuronová síť aktivně působí jako agent, algoritmus strojového učení (konkrétně Deep Q-Learning), který umožňuje síti se učit ze zkušeností, a vizualizační nástroj, který graficky znázorňuje aktivaci jednotlivých neuronů a tok informací mezi vrstvami sítě.

Díky této aplikaci mohou uživatelé pozorovat, jak síť reaguje na různé herní situace, které neurony se aktivují při zpracování vstupu, jak informace putují od vstupní vrstvy k výstupní, a konečně, jak síť vybírá konkrétní akci. Projekt tak poskytuje názorný most mezi abstraktní teorií neuronových sítí a jejich praktickým fungováním, čímž usnadňuje pochopení principů umělé inteligence a reinforcement learningu.

2 Neuronové sítě v reinforcement learningu

2.1 Úvod do neuronových sítí

Umělá neuronová síť je výpočetní model inspirovaný biologickými neuronovými sítěmi se schopností naučit se složité nelineární problémy, které je příliš složité algoritmicky popsat [1] [2]. Na rozdíl od tradičních algoritmů, které vyžadují explicitní programování pravidel, neuronové sítě se učí z dat a zkušeností.

V kontextu reinforcement learningu, který je základem tohoto projektu, neuronové sítě slouží jako aproximátory funkce hodnoty – učí se odhadovat, jak dobrá je konkrétní akce v dané situaci. Lidský mozek řeší podobné úlohy pomocí téměř sta miliard navzájem propojených neuronů [3], které vytváří výpočetní síť obrovské komplexity. Umělé neuronové sítě se snaží tento princip zjednodušeně napodobit.

2.2 Umělý neuron

Základem umělé neuronové sítě je umělý neuron [1]. Funguje analogicky k biologickému neuronu: přijímá signály, zpracovává je a vysílá vlastní výstup.

Do umělého neuronu vstupují reálná čísla $x_1, x_2, \ldots, x_n$, která reprezentují informaci z vnějšího okolí nebo výstupy z jiných neuronů [1].

V případě hry Pong vstupní neurony přijímají informace jako pozice pálek, pozice míčku a jeho rychlost.

Každá ze vstupních hodnot má určitý vliv na výstupní hodnotu neuronu. Tento vliv je vyjádřen jednotlivými váhami $w_1, w_2, \ldots, w_n$, kterými se vstupní hodnoty násobí [1]. Během učení se tyto váhy postupně upravují, aby síť dávala lepší rozhodnutí.

2.3 Matematický model umělého neuronu

Každý umělý neuron provádí matematickou operaci, která simuluje zpracování signálu biologickým neuronem [1] [3]. Nejprve vypočítá vážený součet vstupů:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

kde x_i jsou vstupní hodnoty, w_i příslušné váhy a b je tzv. bias (posun) [1].

Výsledná hodnota je následně upravena aktivační funkcí:

$$y = \sigma(z)$$

Aktivační funkce zavádí do modelu nelinearitu. Bez ní by byla síť schopná reprezentovat pouze lineární závislosti [3].

Mezi běžně používané aktivační funkce patří sigmoid, hyperbolický tangens (tanh) a ReLU [3] [4]. ReLU je dnes nejčastější volbou, protože zrychluje učení a zmírňuje problém mizejícího gradientu [4].

2.4 Struktura neuronové sítě

Neuronové sítě jsou organizovány do vrstev – vstupní, jedné či více skrytých a výstupní vrstvy [1] [3].

- **Vstupní vrstva** přijímá data z prostředí.
- **Skryté vrstvy** vytvářejí postupně složitější reprezentace vstupních dat.
- **Výstupní vrstva** poskytuje výslednou predikci nebo rozhodnutí.

Pokud je síť tvořena více skrytými vrstvami, označuje se jako hluboká neuronová síť (deep neural network) .

2.5 Učení neuronové sítě

Cílem učení je upravit váhy tak, aby síť minimalizovala chybu mezi předpovědí a cílovou hodnotou [1].

Během učení se váhy postupně upravují tak, aby síť dělala správnější rozhodnutí. Tento proces se opakuje mnohokrát během trénování[3] [4].

3 Reinforcement Learning

Reinforcement learning (RL), nebo také Deep Q-learning, je metoda strojového učení, ve které se agent učí interakcí s prostředím a získává zpětnou vazbu ve formě odměny [6].

Reinforcement learning popisuje situaci, kdy se agent učí pomocí pokusů a chyb při interakci s prostředím, za správné akce dostává pozitivní odměnu, za špatné akce negativní odměnu.

Cílem agenta je maximalizovat očekávanou kumulativní odměnu v čase [6].

3.1 Q-learning

Q-learning je algoritmus reinforcement learningu, který se snaží naučit tzv. Q-funkci:

$$Q(s, a)$$

Tato funkce udává očekávanou budoucí odměnu při provedení akce a ve stavu s [6].

Q-learning se snaží naučit, jak dobrá je určitá akce v dané situaci. Agent si postupně ukládá zkušenosti z hry a podle získané odměny upravuje své budoucí rozhodování [6].

3.2 Deep Q-Learning

Aby se síť zlepšovala, potřebuje vědět, jak moc se její odhad lišil od skutečně získané odměny. Čím větší byl rozdíl, tím více síť upraví své váhy. Postupným opakováním tohoto procesu se odhady sítě stávají přesnějšími a agent se chová lépe [4] [5].

Učení by bylo nestabilní, kdyby se síť učila ze situací přesně v tom pořadí, jak nastaly. Proto si agent ukládá proběhlé situace do paměti a učí se z náhodně vybraných dávek. Navíc se používají dvě oddělené sítě: jedna rozhoduje během hry, druhá počítá, jak dobré rozhodnutí bylo. Toto oddělení zabraňuje tomu, aby se síť snažila trefit do cíle, který se sám pohybuje [5].

4 Architektura systému

Vytvořená aplikace je navržena jako modulární systém složený z několika vzájemně spolupracujících částí. Hlavním cílem architektury bylo oddělit jednotlivé funkční komponenty tak, aby bylo možné samostatně řešit herní logiku, proces učení neuronové sítě i její vizualizaci.

Celý systém se skládá ze čtyř hlavních částí:

- herní prostředí
- agent využívající Deep Q-Learning
- herní engine řídící běh aplikace
- vizualizační modul neuronové sítě

Jednotlivé komponenty spolu komunikují prostřednictvím sdíleného stavu hry, který reprezentuje aktuální situaci v herním prostředí. Díky tomuto rozdělení je možné například spustit vizualizaci neuronové sítě v samostatném procesu bez zpomalení samotné hry.

4.1 Reprezentace stavu prostředí

Neuronová síť nepracuje přímo s obrazem hry, ale s numerickou reprezentací herního stavu. Každý stav je popsán šesti hodnotami:

- vertikální pozice levé pálky
- vertikální pozice pravé pálky
- horizontální pozice míčku
- vertikální pozice míčku
- horizontální rychlost míčku
- vertikální rychlost míčku

Tyto hodnoty tvoří vstupní vektor neuronové sítě o velikosti šest. Taková reprezentace výrazně zjednodušuje učící úlohu oproti práci s obrazovými daty a zároveň zachovává všechny informace potřebné pro rozhodování agenta.

4.2 Akční prostor

Agent může v každém kroku provést jednu ze tří akcí:

- pohyb pálky nahoru
- setrvání na místě
- pohyb pálky dolů

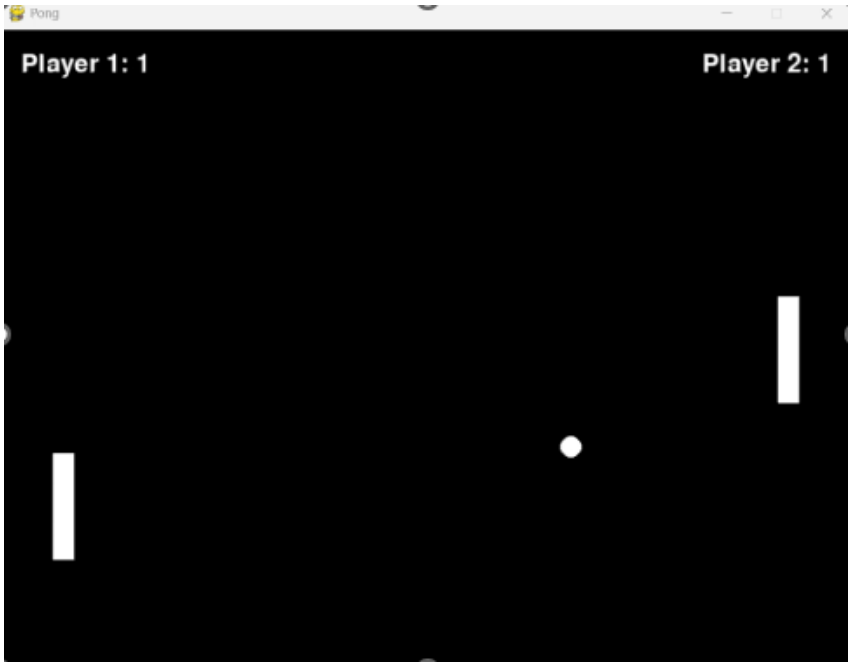
Neuronová síť na výstupu produkuje tři Q-hodnoty, přičemž každá odpovídá jedné možné akci. Vybrána je akce s nejvyšší hodnotou.

5 Aplikace

5.1 Herní prostředí

Herní prostředí je implementováno jako třída `PongEnv`. Ta simuluje pravidla hry Pong, spravuje fyziku pohybu objektů a poskytuje rozhraní kompatibilní s reinforcement learningem.

Na obrázku 1 je ukázka herního prostředí použitého v aplikaci.



Obrázek 1: Herní prostředí Pong použité v aplikaci

Prostředí implementuje základní metody:

- `reset()` – inicializuje novou epizodu hry
- `step(action)` – provede akci agenta a vrátí nový stav
- `render()` – vykreslí aktuální stav hry

Metoda `step()` vrací čtveřici hodnot:

- nový stav prostředí
- odměnu
- informaci o ukončení epizody

- doplňující informace

Odměnová funkce je navržena tak, aby podporovala žádoucí chování agenta. Agent získává odměnu za:

- + 0,5 za trefení míčku pálkou
- + 1 za zaskórování bodu - trefení míčku na soupeřovu stranu
- - 0.001 pokaždé, když se zavolá metoda step()
- negativní odměna, pokud se netrefí * vzdálenost, o kterou se netrefí

5.2 Agent a neuronová síť

Agent je implementován ve třídě `Agent`. Jeho hlavní součástí je hluboká neuronová síť využívající Q-funkci.

Použitá architektura sítě:

- vstupní vrstva: 6 neuronů
- první skrytá vrstva: 128 neuronů
- druhá skrytá vrstva: 64 neuronů
- výstupní vrstva: 3 neurony

Výstupní neurony reprezentují Q-hodnoty jednotlivých akcí. Síť je implementována pomocí knihovny PyTorch.

Pro výběr akcí je použita strategie ε-greedy. S pravděpodobností ε agent zvolí náhodnou akci (explorace), jinak vybírá akci s nejvyšší Q-hodnotou (exploatace).

5.3 Proces učení

Učení probíhá pomocí algoritmu Deep Q-Learning. Během hry agent ukládá své zkušenosti do paměti.

Při trénování jsou náhodně vybírány dávky zkušeností.

5.4 Postupné trénování agentů

Během vývoje aplikace bylo vytvořeno několik verzí agenta, které se postupně lišily způsobem trénování, nastavením odměn a herním prostředím. Cílem bylo postupně zlepšovat schopnost neuronové sítě hrát hru Pong.

Každý agent byl trénován po určitém počtu cyklů. Jeden cyklus představuje úsek hry od odpalu míčku až do okamžiku, kdy míček dopadne na některou ze stran hřiště.

5.4.1 První modely agentů

První vytvořený model (Agent 1) používal neuronovou síť o architektuře 6-128-64-3 a byl trénován po dobu 500 cyklů. Odměna byla nastavena pouze na hodnotu +1 za zaskórování bodu, tedy za situaci, kdy míček dopadl na soupeřovu stranu.

Ukázalo se, že takto natrénovaný agent hru téměř nehraje. Namísto snahy míček odrážet se často míčku spíše vyhýbal, protože systém odměn neposkytoval dostatečnou motivaci k aktivní hře.

Následující modely Agent 2 a Agent 3 byly trénovány po větším počtu cyklů, ale jinak používaly stejné nastavení jako Agent 1. Výsledek byl velmi podobný, agenti se chovali téměř stejně, pouze samotné trénování trvalo výrazně déle.

5.4.2 Změna počátečních podmínek

U modelu Agent 4 byl zachován stejný typ neuronové sítě (6-128-64-3), ale trénování probíhalo po dobu 1000 cyklů a první odpal míčku byl vždy směrem k agentovi. Tím bylo zajištěno, že agent dostane více příležitostí míček zasáhnout.

Tato změna vedla pouze k menšímu zlepšení chování agenta.

5.4.3 Vylepšení systému odměn

Výraznější zlepšení přinesl až model Agent 5. U tohoto agenta byl zaveden podrobnější systém odměn:

- +0,5 za trefení míčku pálkou
- +1 za zaskórování bodu
- -0,001 při každém volání metody `step()`

- negativní odměna při netrefení míčku úměrná vzdálenosti, o kterou se agent netrefil

Díky tomuto systému byl agent motivován nejen ke skórování bodů, ale také k samotnému odrážení míčku a k rychlejšímu dokončení výměny. Výsledkem bylo výrazné zlepšení herního chování.

5.4.4 Trénování bez soupeře

Všechny dosavadní modely byly trénovány proti nepohyblivé pálce. To znamená, že agent se učil pouze reagovat na pohyb míčku, nikoli na chování soupeře.

Po přidání skutečného soupeře se ukázalo, že takto natrénovaní agenti mají s hrou proti aktivnímu protivníkovi problémy. Jejich chování bylo podobné jako u prvního modelu, často se míčku spíše vyhýbali.

5.4.5 Trénování proti soupeři

Model Agent 6 byl prvním agentem, který byl trénován proti aktivnímu soupeři. Používal stejnou architekturu neuronové sítě (6-128-64-3) a byl trénován po dobu 1000 cyklů.

Ukázalo se však, že pokud ani jeden z agentů neumí dobře reagovat na soupeře, samotné společné trénování nevede k výraznému zlepšení.

5.4.6 Postupné zlepšování soupeřů

Nejlepších výsledků bylo dosaženo u modelu Agent 7. Trénování probíhalo v několika kolech, přičemž v každém kole se agent učil proti o něco lepšímu soupeři.

Postup byl následující:

1. Agent byl nejprve trénován proti nepohyblivé pálce.
2. Poté byl přesunut na druhou stranu hřiště a stal se soupeřem nového agenta.
3. Vytvořil se nový agent, který se učil proti tomuto již natrénovanému protivníkovi.
4. Po dokončení trénování byl starší agent odstraněn a proces se opakoval.

Tento postup byl opakován celkem desetkrát. Každý nový agent tak trénoval proti o něco silnějšímu soupeři.

Výsledkem byl dosud nejlepší model, který dokáže při běžné rychlosti hry poměrně dobře konkurovat lidskému hráči. Při vyšší rychlosti hry má agent dokonce výraznou výhodu, protože dokáže reagovat rychleji než člověk.

5.4.7 Druhá generace agentů

V další fázi experimentů byla zkoušena také menší architektura neuronové sítě se skrytými vrstvami o velikosti 16 a 4 neuronů. Předchozí modely přitom používaly vrstvy o velikosti 128 a 64 neuronů.

I když byly tyto menší modely trénovány stejným způsobem jako Agent 7, ukázalo se, že jejich schopnost hrát hru je výrazně horší. Ve většině situací vyhodnocovaly jako nejlepší akci setrvání na místě a míček často vůbec nezasahovaly.

5.4.8 Shrnutí vývoje

Během vývoje agentů byly všechny úspěšné změny zachovány i v následujících verzích. Například pokud byl vylepšen systém odměn nebo způsob trénování, všechny další modely již používaly toto nové nastavení.

Nejlepší výsledky tak dosahuje Agent 7, který kombinuje vylepšený systém odměn, trénování proti postupně silnějším soupeřům a původní větší architekturu neuronové sítě.

5.5 Herní engine

Hlavní běh aplikace řídí modul `game_engine.py`. Ten propojuje herní prostředí, agenta a vizualizační nástroj.

Engine zajišťuje:

- načtení natrénovaného modelu
- řízení herní smyčky
- zpracování vstupu uživatele
- předávání aktuálního stavu neuronové vizualizaci

Aplikace umožňuje několik režimů běhu: hraní hráče proti agentovi, sledování hry dvou agentů nebo klasickou hru pong hráče proti hráči.

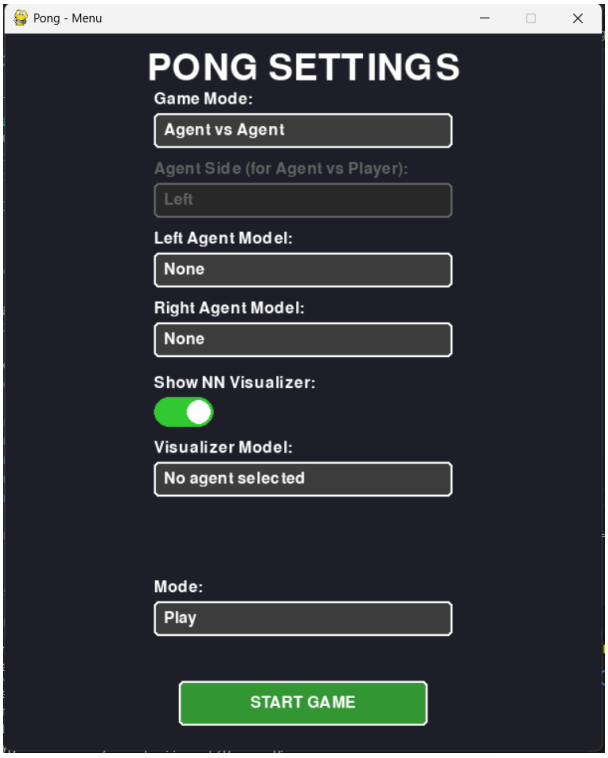
5.6 Uživatelské rozhraní

Před spuštěním samotné hry se zobrazí úvodní menu, které umožňuje nastavit základní parametry aplikace. Uživatel může zvolit herní režim: hru dvou agentů proti sobě, hru hráče proti agentovi nebo klasickou hru dvou hráčů.

Dále je možné vybrat natrénovaný model pro levého a pravého agenta a zapnout nebo vypnout vizualizaci neuronové sítě.

Díky tomuto rozhraní není nutné zasahovat do kódu aplikace, vše potřebné lze nastavit přímo před spuštěním hry.

Na obrázku 2 je ukázka úvodního menu aplikace.



Obrázek 2: Úvodní menu aplikace

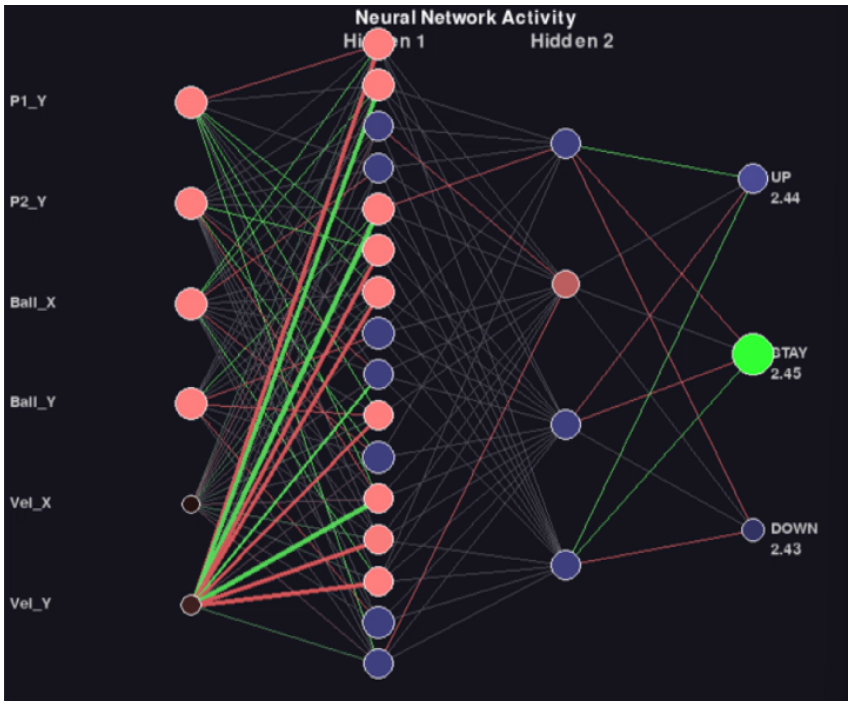
5.7 Vizualizace neuronové sítě

Jednou z klíčových částí projektu je vizualizační modul, který zobrazuje aktivitu neuronové sítě v reálném čase.

Na obrázku 3 je ukázka vizualizace neuronové sítě vytvořené v aplikaci.

Model na obrázku se skládá z šesti vstupních neuronů a třech výstupních, ve skrytých vrstách je potom 16 a 4 neurony. Je to model druhé generace, na vizualizaci je sice

lépe vidět samotné "přemýšlení", ale kvůli malému počtu neuronů takový model téměř nedokáže hrát.



Obrázek 3: Vizualizace neuronové sítě během hry

Vizualizace běží v samostatném procesu, aby nedocházelo ke zpomalení hry. Komunikace mezi hrou a vizualizací probíhá pomocí mechanismu Pipe z knihovny `multiprocessing`.

Vizualizace zobrazuje:

- jednotlivé vrstvy neuronové sítě
- aktivaci neuronů
- sílu a znaménko vah mezi neurony
- aktuálně vybranou akci agenta

Velikost a barva neuronů odpovídá jejich aktivaci, zatímco spojnice mezi neurony znázorňují tok informace sítí. Uživatel tak může přímo sledovat rozhodovací proces neuronové sítě během hry.

6 Instalace a spuštění aplikace

Aplikaci je možné spustit na libovolném zařízení s operačním systémem Windows 11 pomocí příkazové řádky. Následující postup popisuje instalaci všech potřebných nástrojů a spuštění projektu na čistém systému.

6.1 Instalace potřebného softwaru

Nejprve je nutné nainstalovat programovací jazyk Python. Instalace se provede pomocí nástroje `winget` následujícím příkazem:

```
winget install Python.Python.3.13
```

Po dokončení instalace je vhodné zavřít a znovu otevřít příkazovou řádku. Správnou instalaci lze ověřit příkazem:

```
python --version
```

Dále je nutné nainstalovat verzovací systém Git:

```
winget install Git.Git
```

Pro správnou funkčnost některých knihoven (například PyTorch) je potřeba také nainstalovat systémový balíček Microsoft Visual C++ Redistributable:

```
winget install Microsoft.VCRedist.2015+.x64
```

6.2 Stažení projektu

Projekt je uložen na platformě GitHub a lze jej stáhnout pomocí příkazu:

```
git clone https://github.com/Gabi1818/Neural-Network---Pong.git
```

Následně je potřeba přejít do složky projektu:

```
cd Neural-Network---Pong
```

6.3 Instalace knihoven

Projekt využívá externí knihovny, které je nutné nainstalovat pomocí nástroje pip:

```
pip install -r requirements.txt
pip install torch
```

6.4 Spuštění aplikace

Po úspěšné instalaci všech závislostí lze aplikaci spustit příkazem:

```
python game_engine.py
```

Aplikace byla otestována na čisté instalaci systému Windows 11.

7 Možná rozšíření projektu

Současná implementace funguje spolehlivě pro hru Pong, má však několik omezení, která by bylo možné v budoucnu odstranit.

Architektura neuronové sítě je v současné verzi pevně daná, což znamená, že síť má vždy dvě skryté vrstvy a počet neuronů v každé z nich musí být nastaven přímo v kódu. Pokud by si uživatel chtěl vyzkoušet jinou velikost sítě, musel by zasáhnout do zdrojového kódu. Přirozeným rozšířením by bylo přidat možnost nastavit počet neuronů přes konfigurační soubor nebo přímo v uživatelském rozhraní aplikace. S tím souvisí i to, že uložený natrénovaný model v současné době neobsahuje informaci o tom, kolik neuronů ve skrytých vrstvách má, při načítání modelu je tedy nutné zadat stejnou architekturu, jaká byla použita při trénování, jinak model nepůjde načíst.

Zajímavou oblastí pro experimenty je také systém odměn. Aktuálně agent dostává odměnu například za trefení míčku nebo za získání bodu. Bylo by možné vyzkoušet různé kombinace odměn, například penalizovat zbytečný pohyb pálky a sledovat, zda taková změna vede k lepšímu nebo horšímu výsledku. Tyto experimenty by mohly být zajímavým rozšířením i jako samostatné cvičení pro studenty, kteří by aplikaci používali ve výuce.

Do budoucna by bylo možné aplikaci rozšířit i o podporu dalších her s podobnou strukturou, nebo umožnit uživateli měnit parametry učení, například míru náhodného průzkumu a sledovat jejich vliv na výkon agenta v reálném čase.

8 Závěr

Cílem této práce bylo vytvořit aplikaci, která umožňuje názorně sledovat fungování jednoduché neuronové sítě při řízení počítačové hry. V rámci projektu byl navržen a implementován systém kombinující herní prostředí, agenta využívajícího algoritmus Deep Q-Learning a vizualizační nástroj zobrazující vnitřní stav neuronové sítě během rozhodování.

Součástí projektu bylo využití herního prostředí Pong, které slouží jako modelové prostředí pro experimenty s reinforcement learningem, které bylo převzato z veřejně dostupného projektu na platformě GitHub a následně upraveno a integrováno do vlastní aplikace.

Prostředí poskytuje agentovi informace o aktuálním stavu hry prostřednictvím definovaného stavového prostoru a umožňuje provádět akce odpovídající pohybu herní pálky. Na základě těchto vstupních dat může neuronová síť vyhodnocovat jednotlivé situace a rozhodovat o dalším pohybu agenta.

Pro řízení agenta byla implementována hluboká neuronová síť využívající algoritmus Deep Q-Learning. Síť se během trénování učí odhadovat Q-hodnoty jednotlivých akcí a postupně zlepšuje své rozhodování na základě získaných zkušeností.

Důležitou součástí projektu je vizualizační modul, který umožňuje sledovat aktivaci neuronů a tok informací mezi vrstvami sítě. Díky tomu může uživatel pozorovat, jak neuronová síť reaguje na aktuální stav hry a jakým způsobem dochází k výběru výsledné akce.

Výsledná aplikace může sloužit jako výukový nástroj pro základní pochopení principů neuronových sítí a reinforcement learningu.

Reference

[1] KRATOCHVÍL, Tomáš. Neuronové sítě a jejich aplikace. Online, Bakalářská práce, vedoucí doc. Mgr. Dušan Bednařík, Ph.D. Hradec Králové: Univerzita Hradec Králové Přírodovědecká fakulta, Katedra matematiky, 2020. Dostupné z: https://theses.cz/id/9knlk7/STAG93390.pdf. [cit. 2026-03-09].

[2] Neuronové sítě. Online. Umíme to Informatika. Dostupné z: https://www.umimeinformatiku.cz/cviceni-neuronove-site-4-trida. [cit. 2026-03-09].

[3] VOLNÁ, RNDr. PaedDr. Eva, PhD. Neuronové sítě 1. Online, Studijní materiály pro distanční kurz. Ostrava: Ostravská univerzita v Ostravě, 2008. Dostupné z: https://web.osu.cz/~Volna/Neuronove_site_skripta.pdf. [cit. 2026-03-09].

[4] LATYPOVA, Dina. NEURAL NETWORKS USING FOR HANDWRITTEN NUMBERS RECOGNITION. Online, Master's Thesis, vedoucí Ing. Karel Frajták, Ph.D. Prague: Czech Technical University in Prague Faculty of Electrical Engineering Department of Computer Science, 2020. Dostupné z: https://dspace.cvut.cz/server/api/core/bitstreams/82c8d9b7-88fc-4737-b99f-2a54d289c88a/content. [cit. 2026-03-09].

[5] RYTÍŘ, Bc. Martin. FYZIKÁLNĚ INFORMOVANÉ NEURONOVÉ SÍTĚ. Online, Diplomová práce, vedoucí Ing. Petr Hadraba, Ph.D. Brno: Vysoké učení technické v Brně, 2025. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=283108. [cit. 2026-03-09].

[6] SUTTON, Richard a BARTO, Andrew. Reinforcement learning. Online. 2nd Edition. England: The MIT Press, 2018, 2020. Dostupné z: http://incompleteideas.net/book/RLbook2020.pdf. [cit. 2026-03-09].

Seznam obrázků