VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POSILOVANÉ UČENÍ PRO HRANÍ ROBOTICKÉHO FOTBALU

REINFORCEMENT LEARNING USED FOR PLAYING ROBOTIC SOCCER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **HYNEK BOČÁN**
AUTHOR

VEDOUCÍ PRÁCE **Doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Bočán Hynek**
Obor: Informační technologie
Téma: **Posilované učení pro hraní robotického fotbalu**
Reinforcement Learning for RoboCup
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s metodami hraní her s využitím posilovaného učení - viz např. https://www.cs.utexas.edu/~pstone/research.shtml
2. Prostudujte podmínky a omezení prostředí definovaného v rámci soutěže RoboCup 3D Soccer Simulation League
3. Implementujte herní strategii založenou na posilovaném učení a nezbytná rozhraní pro zařazení do soutěže
4. Vyhodnoťte vytvořený systém srovnáním s ostatními účastníky
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky

Literatura:

- dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Funkční prototyp řešení

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá tvorbou umělé inteligence schopné ovládat robotického hráče fotbalu simulovaného v prostředí SimSpark. Vytvořená umělá inteligence rozšiřuje již hotového agenta, který poskytuje implementaci základních dovedností jako je orientace na hřišti, pohyb v osmi směrech a nebo driblování s balonem. Umělá inteligence se stará o výběr nejvhodnější akce v závislosti na situaci na hřišti. Pro její implementaci byla použita metoda posilovaného učení - Q-learning. Pro výběr nejlepší akce je situace na hřišti převedena do formy 2D obrazu s několika rovinami. Tento obraz je následně analyzován hlubokou konvoluční neuronovou sítí implementované pomocí C++ knihovny DeepCL.

Abstract

Goal of this thesis is creation of artificial intelligence capable of controlling robotic soccer player simulated in SimSpark environment. Agent created is expanding capabilities of existing third party agent which provides set of basic skills such as localization on the field, dribbling with the ball and omnidirectional walk. Responsibility of the created agent is to pick the best action based current state of the game. This decision making was implemented using reinforcement learning and its method Q-learning. State of the game is transformed into 2D picture with several planes. This picture is then analyzed using deep convolution neural network implemented using C++ and DeepCL library.

Klíčová slova

Strojové učení, posilované učení, hluboké neuronové sítě, konvoluční neuronové sítě, Q-learning, robotický fotbal, RoboCup

Keywords

Machine learning, reinforcement learning, deep neural networks, convolution neural networks, Q-learning, robotic soccer, RoboCup

Citace

BOČÁN, Hynek. *Posilované učení pro hraní robotického fotbalu.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Pavel Smrž, Ph.D.

Posilované učení pro hraní robotického fotbalu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Pavla Smrže. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Hynek Bočán
17. května 2017

Obsah

Úvod

Možnosti uplatnění umělé inteligence se neustále rozrůstají a v jistých úlohách jako například rozpoznánávání psaného textu nebo mluveného slova už se tato technologie stala standardně používaným nástrojem. Ve zmíněných oblastech vynikají zejména metody strojového učení s učitelem. Základním principem těchto metod je znalost správné hodnoty výstupu pro každý tréninkový vstup a poskytnutí této informace agentovi při jeho trénování. Negativem těchto metod je nižší flexibilita a extrémní náročnost na kvantitu správně označkovaných tréninkových dat. Alternativami, které tyto nevýhody eliminují, jsou strojové učení bez učitele a posilované učení.

Při použití posilovaného učení není nutné znát hodnotu správného výstupu pro konkrétní vstupní data. Agent posilovaného učení metodou pokusu a omylu zkoumá prostředí do kterého byl umístěn. Za každou akci, kterou vykoná obdrží pozitivní či negativní odměnu. Na základě této odměny posiluje chování vedoucí k co nejvyšší odměně. Díky této vlastnosti je možné použít umělou inteligenci i ve velice dynamických prostředích, například ve hrách.

Použití umělé inteligence pro hraní her je již známé dlouho, ale v minulosti se používala zejména technika prohledávání stavového prostoru hry hrubou silou. Příkladem může být počítač Deep Blue od společnosti IBM, který v roce 1997 porazil v šachu úřadujícího světového velmistra Garry Kasparova[2]. Díky novým přístupům a použití hlubokých neuronových sítí je ale nyní možné použití umělé inteligence i pro hraní her vyžadujících vzhledem k rozsáhlosti stavového prostoru i jistou míru intuice. Velmi známým příkladem z poslední doby je například vítězství umělé inteligence nad lidským hráčem ve hře Go v roce 2016[12].

Cílem této práce je vytvoření agenta řídícího strategického uvažování hráče robotického fotbalu, využitím metody posilovaného učení Q-learning. Složitost funkce, která agentovi uděluje odměnu za jeho chování je postupně zvyšována, což vede k osvojení složitějších vzorců chování. Zároveň je agentovi poskytováno stále více informací o prostředi, ve kterém se nachází. V pozdějších fázích experimentů jsou agentovi také udělovány částečné odměny, což vede k výraznému zrychlení učícího procesu.

Kapitola 1

Umělá inteligence

"Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který, kdyby ho dělal člověk, bychom považovali za projev jeho inteligence."[8]

Samotná myšlenka inteligentních strojů je velice stará. Například měděný obr Tálos, který měl za úkol ochraňovat pobřeží Kréty. Tato legenda pochází již z dob antického Řecka. Dalším příkladem může být legenda o Golemovi z Prahy. Golem se oživoval vložením šému, poslouchal člověka, jenž ho oživil, a byl schopný vykonávat práci. Toto chování jej staví téměř na úroveň dnešních průmyslových robotů.

Reálně se však první stroje projevující jistou úroveň inteligence začali objevovat až s rozvojem počítačových věd v polovině 20. století. Na první jednoduchý model umělého neuronu spolu s myšlenkou propojení těchto neuronů do sítě v podobě elektronického mozku lze narazit v práci Warrena S. McCullocha a Waltera Pittse "A logical calculus of the ideas immanent in nervous activity"z roku 1943[11].

V roce 1950 pak britský matematik Alan Turing definoval tzv. Turingův test[7]. Cílem testu je prověřit, jestli stroj nebo počítač opravdu projevuje inteligenci. Podle Touringa lze inteligentním stroj prohlásit, pokud nezávislý pozorovatel při zkoumání písemné komunikace člověka s daným systémem nepozná, která z komunikujících stran je člověk a která počítač.

V poslední dekádě umělá integence zažívá rychlý rozvoj a začíná pronikat do mnoha nových oblastí od překládání textů do jiných jazyků až po autonomní automobily.

1.1 Strojové učení

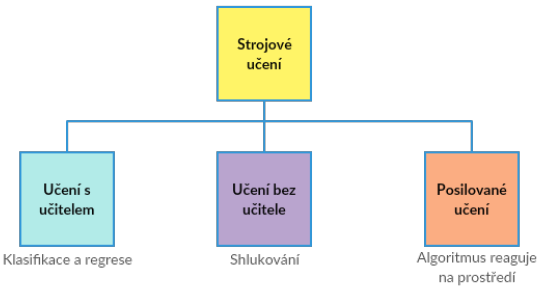
Jedním z nejpoužívanějších metod vytváření umělé inteligence je strojové učení. Tento přístup spočívá v použití algoritmů simulujích učící proces. Základem agenta je matematický model, jehož parametry se upravují tak, aby se agent co nejlépe přizpůsobil problému, který má řešit.

Podle toho, zda je pro natrénovaní agenta potřebné znát správné hodnoty výstupu pro tréninkové vstupy, rozlišujeme metody strojového učení na učení s učitelem, učení bez učitele a posilované učení. Graficky je toto rozdělní znázorněno na obrázku 1.1.

Učení s učitelem vyžaduje sadu trénovacích dat, u které je nutné znát pro každý vstup odpovídající hodnotu výstupu. Agent následně využije trénovací data k upravení parametrů vnitřního modelu a po natrénování je schopen určit výstup pro neznámý vstup.

Pokud nejsou známy správné hodnoty výstupu nebo je cílem najít nové závislosti mezi vstupními daty, nabízí se učení bez učitele. V tomto případě si agent vytváří vlastní re-

Obrázek 1.1: Dělení strojového učení



prezentaci závislotí mezi daty a následně shlukuje vstupy do skupin na základě podobnosti příznaků.

Posledním z přístupů strojového učení je posilované učení. Tato metoda také nevyžaduje znalost správného výstupu pro určitý vstup, ale zároveň poskytuje agentovi zpětnou vazbu za jeho vykonané akce. [6]

Každá z metod strojového učení se hodí k řešení jiného typu problému a nedá se říct, že některý z přístupů by byl obecně lepší než jiný. Pro výběr správné metody je vždy potřeba zvážit v jakém formátu jsou dostupná vstupní data a jestli lze tato data jednoduše rozdělit do konkrétních kategorií.

S využitím strojového učení můžeme řešit tyto 4 základní typy úloh.

1. **Klasifikace** Úkol klasifikace spočívá ve vytvoření prediktivního modelu, který mapuje vstupních objekty do předem známých skupin. Vstupní objekt je popsán příznaky. Agent na základě znalosti příznaků tréninkových objektů zařadí neznámý objekt do jedné z naučených skupin.

2. **Regrese** Cílem regrese je vytvoření prediktivního modelu, který odhaduje hodnotu na výstupu pro určená vstupní data. Výstup má v této úloze podobu spojité veličiny.

3. **Shlukování** Cílem shlukové analýzy je objevit v datech skupiny s podobnými vlastnostmi. Agent nedostává žádnou informaci o požadovaném výstupu a vytváří si vlastní shluky vstupních objektů na základě podobnosti jejich příznaků.

4. **Reakce na prostředí** V této úloze se agent přizpůsobuje prostředí do kterého byl umístěn. Za své akce dostává pozitivní či negativní odměnu. Cílem je naučení takového chování, které maximalizuje dosaženou odměnu.

Z výše uvedených úloh se pro aplikaci na problém strategického uvažování v robotickém fotbalu nejlépe hodí reakce na prostředí a tudíž posilované učení. Při hře existuje spousta situací, které lze pouze s velkými obtížemi označkovat a zařadit do nějaké kategorie. Použití například učení s učitelem by tak bylo problematické.

1.2 Posilované učení

Posilované, někdy také zpětnovazebné učení, je metodou, která napodobuje jeden ze základních učebních procesů živých tvorů. Tato metoda spočívá v udělování pozitivní odměny za chování, které chceme podpořit a negativní odměny za chování, jež chceme vytěsnit. S posilovaným učením se setkal každý, kdo někdy vlastnil domácího mazlíčka. Pokud například

odměníte psa pamlskem pokaždé, když si lehne vedle pohovky a naopak, okřiknete ho když vyskočí na pohovku, je velice pravděpodobné, že si po uplynutí určitého počtu opakování začne lehat pouze vedle pohovky i když by mnohem raději ležel nahoře.

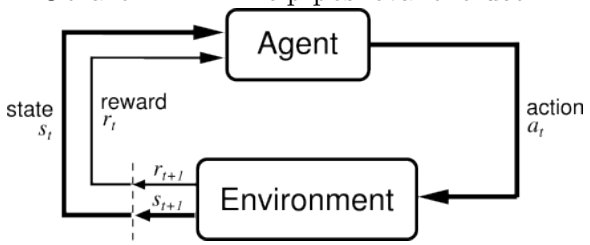
Na podobném princupu funguje posilované učení i u strojů. Agent posilovaného učení interaguje se svým okolím pomocí dostupných akcí a pozoruje důsledek svého počínání. Zpětná vazba za vykonané akce je mu udělována ve formě pozitivní nebo negativní odměny. Tato odměna navíc může přicházet se zpožděním. Na základě udělené odměny agent posiluje chování vedoucí k pozitivní odměně a naopak omezuje chování vedoucí k trestu.

Základní systém posilovaného učení se skládá z následujících částí:

- množina stavů prostředí S
- množina možných akcí agenta a
- pravidla přechodů mezi stavy
- pravidla určující odměnu po přechodu do nového stavu
- pravidla určující cíle agenta

Tok informací v systému posilovaného učení znázorněn na obrázku 1.2.

Obrázek 1.2: Princip posilovaného učení



Zdroj

Posilované učení se výborně hodí i k řešení problémů, u nichž může malá krátkodobá ztráta vést k velkému zisku z dlouhodobého hlediska. Díky tomu se tato metoda osvědčila při řešení mnoha různorodých úloh od ovládání robotů až po hraní her Backgammon nebo Go. Díky uvedeným vlastnostem se také skvěle hodí k aplikaci na problém strategického uvažování v robotickém fotbalu.

1.2.1 Problém objevování versus využívání

Jeden z hlavních problémů při použití posilovaného učení se nazývá objevování versus využívání. Podstatou tohoto problému je stav, kdy se agent naučí v konrétní situaci určitý vzorec chování, který vede k jisté pozitivní odměně. Následně přestane v daném situaci zkoušet nové vzorce chování, které by mohly vést k odměně vyšší. Tato situace se označuje jako využívání. S použitím metafory z úvodu této kapitoly. Pes se již naučil lehat si vedle pohovky a ví, že za toto chování dostane pamlsek a tohoto chování se drží, jelikož vede k zaručené odměně. Nicméně nikdy nezjistí, že pokud by vyskočil na křeslo, dostal by také pamlsek a navíc by se mu leželo pohodlněji. Častým řešením tohoto problému je nastavení agenta tak, aby i přes znalost akce, která v daném stavu vede k pozitivní odměně zkoušel i jiné akce a až později začal upřednosňovat tu, která vede k jisté odměně. Pokud je tato problematika zanedbána může to vést k neuspokojivým výsledkům a zejména neobjevení nejlepšího možného řešení.[13]

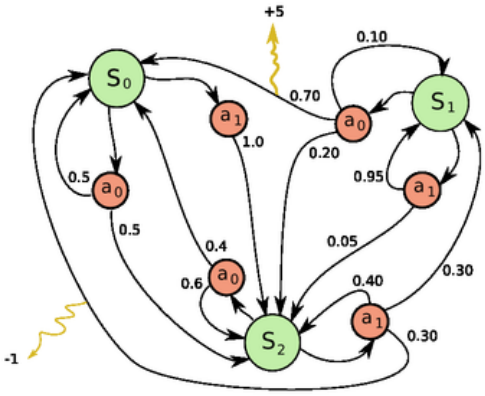
1.2.2 Markovův rozhodovací proces

"Markovské rozhodovací procesy poskytují matematický rámec pro modelování rozhodování v situacích, kdy jsou výsledky zčásti náhodné a zčásti pod kontrolou uživatele."[3]

Pokud v systému posilovaného učení závisí budoucí stav prostředí pouze na stavu aktuálním, můžeme rozhodování tohoto agenta popsat pomocí Markovských rozhodovacích procesů. Markovské rozhodovací procesy jsou použity jako teoretický základ mnoha metod posilovaného učení.

Jednodušší Markovské rozhodovací procesy se dají znázornit i ve formě grafu. Ukázkový Markovský proces obsahující tři stavy a dvě možné akce je uveden na obrázku 1.3.

Obrázek 1.3: Provedení akce a_0 ve stavu S_1 vede k udělení odměny o velikosti +5. Po provedení akce a_1 ve stavu S_2 obdrží agent odměnu o velikosti +1.



Zdroj

Robotický fotbal lze popsat pomocí Markovských procesů. Následující situace na hřišti je vždy závislá pouze na aktuálním stavu. Počet situací, které na hřišti mohou nastat je konečný. Zároveň každý z agentů má pouze omezený počet akcí, kterými může své okolí ovlivnit. Další vývoj situace na hřišti tedy může jeden agent ovlivnit pouze částecně. Jistou výzvou v tomto směru je to, že v prostředí, do kterého byl agent umístěn, se pohybuje až 20 dalších autonomních agentů. Problematikou aplikace Markovovych procesů na multiagentní systémy ze zabývá ve své práci Michael L. Littman[10]. Pokud zkoumaný systém splňuje podmínku konečnosti a lze popsat pomocí Markovských procesů můžeme pro hledání optimálního chování v tomto systému použít metodu Q-learning.

1.2.3 Q-learning

Bezmodelová metoda posilovaného učení Q-learning funguje následovně. Agent je umístěn do neznámého prostředí, které se může nacházet v některém z množiny stavů S. Pro každý prvek množiny S jsou určeny akce A, které má agent dovoleno v tomto stavu vykonat. Každá z akcí má pro určený stav různou hodnotu Q. Čím vyšší je hodnota akce v daném stavu, tím větší je šance, že si agent tuto akci vybere.

Při inicializaci může být pro jednoduchost všem akcím přidělena ve všech stavech stejná hodnota, například 0. Toto nastavení vede k tomu, že jsou na začátku akce vybírány náhodně. Agent je umístěn do prostředí, vykoná svou první akci a pozoruje výši udělené odměny. Pokud akce vedla k pozitivní odměně, je zvýšena její hodnota pro daný stav.

Podobně pokud nastal opačný případ a agent obdržel negativní odměnu. Hodnota akce v daném stavu je snížena a příště až se bude agent nacházet v tomto stavu, je šance, že si vybere negativně hodnocenou akci nižší.

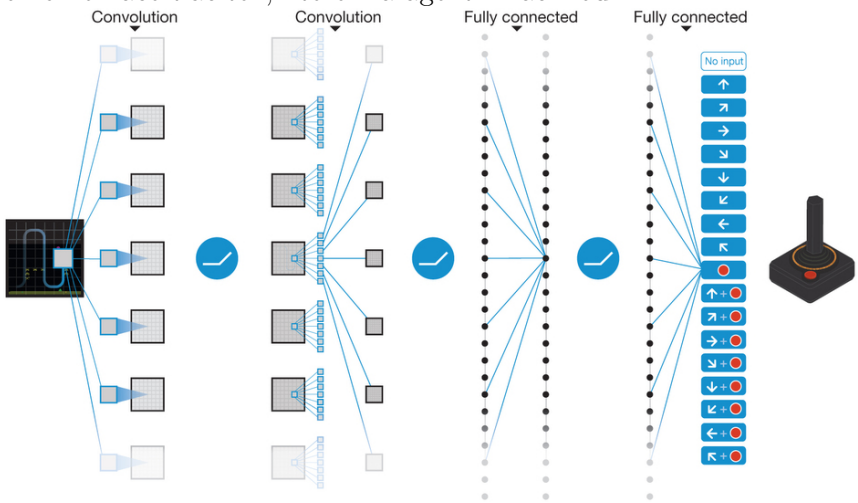
Tento postup může být popsán následujícím pseudokódem [5]:

```
Initialize Q(S, a)
Repeat (for each epizode):
    Initialize S
    Repeat (until epizode ends):
        Choose a with highest Q based on current state S
        Take action a and observe r and S'
        Update Q = Q(S, a) + LR [ r + DF maxR' - Q(S, a) ]
        Update S = S'
```

Dalšími parametry použitými v pseudokódu jsou:

- LR - Learning Rate (rychlost učení), kontanta s hodnotou v intervalu $< 0, 1 >$ udávající rychlost učení. Pokud je rovna 0, hodnoty akcí nejsou nikdy upraveny a nedochází k učení.
- DF - Discount Factor (faktor stárnutí), kontanta s hodnotou v intervalu $< 0, 1 >$. Slouží pro zvýhodnění současných odměny v porovnání s budoucí odměnou. Udává jakou váhu přikládá agent budoucím odměnám.
- maxR - maximální výše odměny Q v následujícím stavu při provedení nejvýhodnější akce pro tento stav.

Obrázek 1.4: Q-learning ve spojení s konvoluční neuronovou sítí. Výsledkem zpracování je tlačítko nebo kombinace tlačítek, které má agent zmáčknou



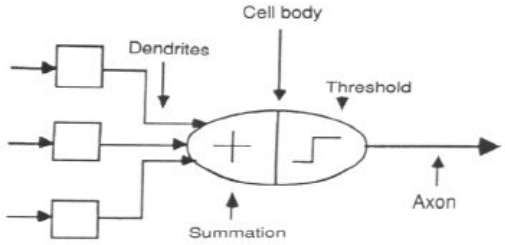
Zdroj

Velkou výhodou metody Q-learning je fakt, že agent nepotřebuje znát přesný model prostředí ve kterém se nachází. Díky této vlastnosti lze ve spojení s konvolučními sítěmi Q-learning uplatnit na hraní her, při kterém jako vstup poskytneme neuronové síti pouze hodnoty všech pixelů na herní obrazovce. Tato možnost uplatnění je znázorněna na obrázku 1.4.

1.3 Neuronové sítě

Umělé neuronové sítě jsou model, silně inspirovaný fungováním nervového systému a tím, jakým způsobem zpracovává informace. Základním stavebním prvkem nervového systému je neuron. Součástí biologického neuronu jsou krátké dostředivé výstupky dendrity, kterými neuron přijímá vstupní informace, a dlouhý odstředivý výstupek axon, který slouží jako výstup neuronu. Základním prvkem neuronových sítí je model umělého neuronu někdy také označován jako perceptron, jehož schematické znázornění ukazuje obrázek 1.5.

Obrázek 1.5: Model umělého neuronu



Zdroj

Neuronová síť se skládá z několika vzájemně propojených neuronů. Každý ze vstupů neuronu má přiřazenou svou váhu, jejímž upravováním dochází k přizpůsobování sítě konkrétnímu problému čili k učení. Jaká hodnota se bude nacházet na výstupu neuronu závisí na použité aktivační funkci a vážené sumě hodnot na jeho vstupech. Rovnice 1.1 popisuje základní model neuronu využívající aktivační funkci signum. Pokud vážená suma vstupních hodnot přesáhne prahovou hodnotu, je na výstupu tohoto neuronu 1 v ostatních případech 0. Výstup jednoho neuronu může sloužit jako vstup dalším neuronům v takovém případě se jedná o hlubokou neuronovou síť.

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases} \tag{1.1}$$

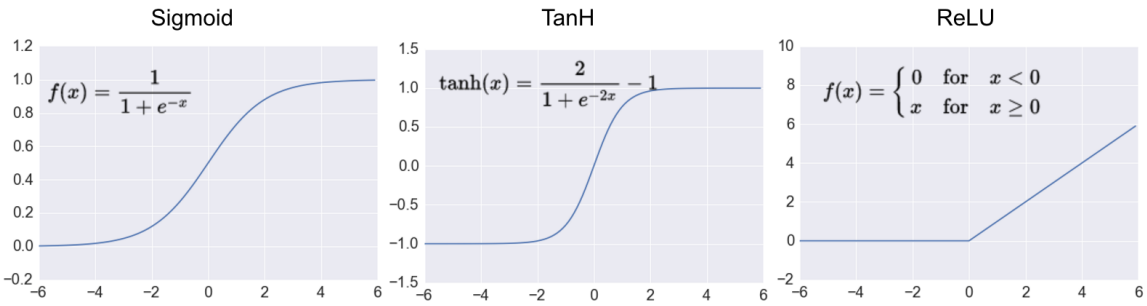
1.3.1 Aktivační funkce

Přesnou hodnotu výstupu neuronu pro určitý vstup udává aktivační funkce. Nejjednodušší aktivační funkcí je funkce signum. Tato funkce udává přesnou prahovou hodnotu při které se skokově změní hodnota na výstupu neuronu. Přehled nejpoužívanějších aktivačních funkcí, které se využívají v konvolučních neuronových sítích ukazuje obrázek 1.6.

Výběr správné aktivační funkce není jednoduchý a mnohdy se provádí pouze na základě přechozích zkušeností nebo experimentováním. Jednou z nejčastěji používaných funkcí v konvolučních neuronových sítích je funkce ReLU. Tato jednoduchá funkce je popsána rovnicí 1.2.

$$f(x) = max(0, x) \tag{1.2}$$

Obrázek 1.6: Zleva funkce sigmoid, hyperbolický tangens, a ReLU (rectified linear unit)



Zdroj

1.3.2 Hluboké neuronové sítě

Klasická neuronová síť se skládá ze vstupní, skryté a výstupní vrstvy. Jako hluboká se neuronová síť označuje pokud obsahuje více než jednu skrytou vrstu a tedy se skládá z více než 3 vrstev.

Hluboké neuronové sítě umožňují zpracovávat vstupní data s několika úrovněmi abstrakce v závislosti na hloubce použité sítě. Síla hlubokých neuronových sítích spočívá v tom, že se každá skrytá vrstva dostává na vstupu odlišná data. Výstup jedné vrstvy je použit jako vstup další vrstvy. S originálními vstupními daty tedy pracuje pouze první vrstva v pořadí, každá další má už na vstupu data zpracovaná předchozí vrstvou. Každá ze skrytých vrstev navíc může používat odlišnou aktivační funkci. S každou další vrstvou se rozeznávané příznaky stávají komplexnějšími, avšem také se stává složitějším učení neuronové sítě zejména pak výpočet nových vah.

1.3.3 Konvoluční sítě

Pokud bychom pro analýzu obrázku z datasetu CIFAR-10, který má velikost 32x32 px a 3 kanály pro barvu, chtěli použít obyčejnou neuronovou síť znamenalo by to, že i pro takto malý obrázek by každý z neuronů měl 3072 vstupů[1]. Z tohoto důvodu jsou klasické neuronové sítě nepoužitelné pro zpracování větších obrazů. Jedním z řešení tohoto problému jsou konvoluční sítě.

Konvoluční sítě patří do skupiny hlubokých neronových sítí a v současné době patří k nejslibnější technologii při zpracovávání obrazových dat. Velkou výhodou konvolučních sítí je, že vyžadují minimální úpravy vstupních obrazů, vstupem jsou jednotlivé pixely zkoumaného obrazu[9]. Týmy používající konvoluční sítě pravidelně stanovují nové rekordy v benchmarcích přesnosti rozpoznávání obrazu například ImageNet[4].

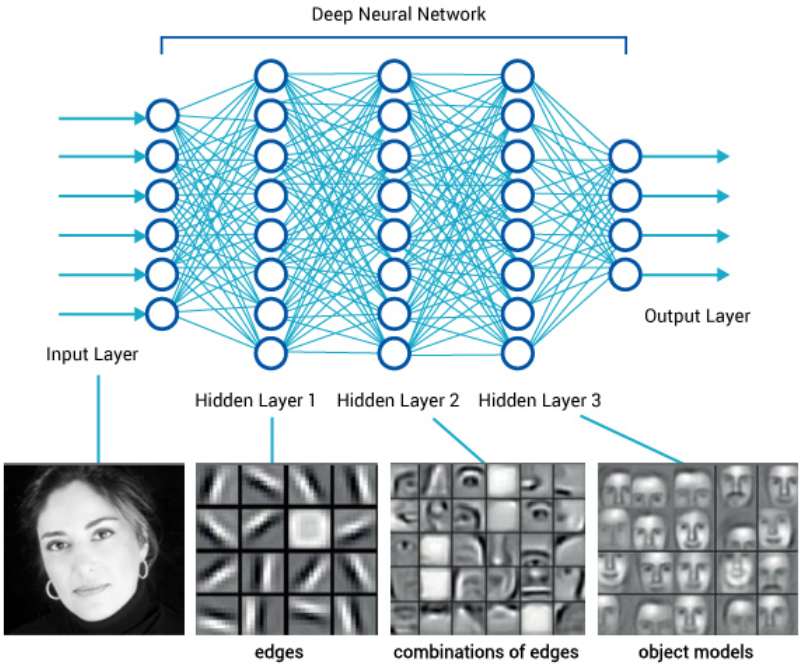
Konvoluční sítě zpracovávají obraz po menších částech ze kterých se snaží extrahovat příznaky pomocí filtrů. Použití konvolučních filtrů pro extrakci různých příznaků je ukázáno na obrázku1.7. To jak mohou reálně vypadat příznaky konvoluční sítě ukazuje obrázek 1.8

Obrázek 1.7: Příklady konvolučních filtrů pro detekci různých vlastností

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Zdroj

Obrázek 1.8: Příklad extrakce příznaků konvoluční neuronovou sítí



Zdroj

Kapitola 2

Implementace

Cílem implementace bylo vytvoření agenta ovládajícího robota Nao v simulačním prostředí SimSpark[1].

Vytvořený agent vznikl propojením dvou nezávislých programů. První obstarává zpracování informací o prostředí a také vykonává jednotlivé akce jako je chůze a kopání do balónu. Druhý převádí získané informace o situaci na hřišti do formy obrazu s několika rovinami. Vytvořený obraz je tímto programem dále zpracováván hlubokou konvoluční neuronovou sítí a následně vrací prvnímu programu akci, kterou má vykonat.

Pro obě části byly použity jako základ již existující knihovny. Jelikož projektů zabývajících se tvorbou agentů robotického fotbalu je řádově méně než těch z oblasti umělé inteligence byla volba programovacího jazyka ovlivněna zejména existencí kvalitního základního fotbalového agenta.

Nakonec byl vybrán agent vytvořený University of Texas v Austinu, utaustinvilla3d[2]. Tento agent je napsán v jazyce C++ a byl vydán proto, aby snížil vstupní bariéru pro další zájemce o simulovaný robotický fotbal. Poskytuje hotovou implementaci všech základních dovedností potřebných pro hráče - chůzi, orientace na hřišti, zvednutí po pádu, driblování s míčem a mnoho dalších. Odstraněny z něj bylo pouze strategické uvažování čili ta část programu, která rozhoduje o tom, jaká akce se mám v daný moment vykonat.

Knihovny pro analýzu situace na hřišti byly vybírány již s přihlédnutím k pozdější nutnosti provázání s agentem a tudíž byly brány v potaz pouze ty napsané v C++. S přihlédnutím k jednoduchosti implementace a existujícímu příkladu Q-learningu byla nakonec vybrána knihovna DeepCL[3].

V první fázi implementace jsou prováděny experimenty mimo prostředí SimSpark. V těchto experimentech je neuronové síti ke zpracování předáván jako vstup obraz o jedné až třech rovinách. Jednotlivé roviny obrazu reprezentují například pozici agenta na hřišti, jeho orientaci nebo pozici míče na hřišti. Pro zrychlení experimentování v první fázi agent pohybuje v extrémně zjednodušeném 2D prostředí.

Ve druhé fázi dojde k propojení natrénované neuronové sítě z první fáze s fotbalovým agentem. Tato fáze vyžaduje propojení obou programů, jelikož je třeba inicializovat neuronovou síť v programu hráče. Dále je potřeba v programu hráče implementovat akce, které mapují ty dostupné v experimentech první fáze.

[1] http://simspark.sourceforge.net/
[2] https://github.com/LARG/utaustinvilla3d
[3] https://github.com/hughperkins/DeepCL

V poslední, třetí fázi, se pokusím propojit učební scénáře z knihovny DeepCL se simulačním serverem. V případě úspěšného propojení bude agent schopný se učit z reálně odehraných her proti skutečným týmům.

2.1 RoboCupSoccer

RoboCupSoccer je sportovní liga robotického fotbalu jak v reálném tak simulovaném prostředí, hraná pod hlavičkou mezinárodní výzkumné a vzdělávací organizace RoboCup[4]. Cílem organizace je podpořit výzkum umělé inteligence poskytnutím standardizovaných problémů, k jejichž řešení lze přistupovat mnoha různými způsoby a s využitím široké palety technologií. Fotbal byl vybrán vzhledem ke všeobecné znalosti jeho pravidel, popularitě a v neposlední řadě i složitosti možných herních strategií. Umožňuje testovat cokoliv od jednoduchých algoritmů pro zpracování vstupních dat a určení pozice hráče na hřišti, až po složité strategické uvažování založené například na posilovaném učení.

Hlavním dlouhodobým cílem je, aby do roku 2050 tým plně autonomních robotických hráčů vyzval právě úřadujícího vítěze světového šampionátu ve fotbalu a porazil jej.

V počátcích se simulační fotbalová liga hrála pouze ve 2D prostředí s velice omezeně implementovanými fyzikálními jevy a naprostým zjednodušením hráčů pouze na kruhy. S příchodem 3D platformy v roce 2003 [5] se simulace přenesla do mnohem realističtějšího prostředí simulátoru SimSpark. S touto změnou také simulovaní roboti začali reflektovat existující robotické předlohy. Prvním použitým modelem byl robot Fujitsu HOAP-2, který od roku 2008 nahradil model Aldebaran Nao, používaný dodnes.

2.1.1 SimSpark

Multiagentní simulátor SimSpark vznikl pod hlavičkou organizace RoboCup. Pro realistické simulování pohybu modelů, detekci kolizí a tření používá knihovna Open Dynamics Engine2(ODE)[6]. Tato knihovna také poskytuje prostředky pro pokročilé modelování motory ovládaných kloubů agentů.

Jedná se o serverovou aplikaci, se kterou agenti komunikují pomocí UDP nebo TCP socketů. Z toho vyplývá, že pro tvorbu agenta lze použít libovolný programovací jazyk podporující alespoň jeden z těchto socketů. K samotnému serveru byla vytvořena i aplikace monitor, sloužící k vizualizaci situace na herní ploše. Náhled výchozího monitoru 2.1.

Simulace probíhá v diskrétním čase v cyklech o délce 20 ms. Většina pravidel hry je rozhodována automaticky. složitější situace ale stále vyžadují manuální zapojení lidského rozhodčího.

2.1.2 Specifikace hřiště, agenta a hry

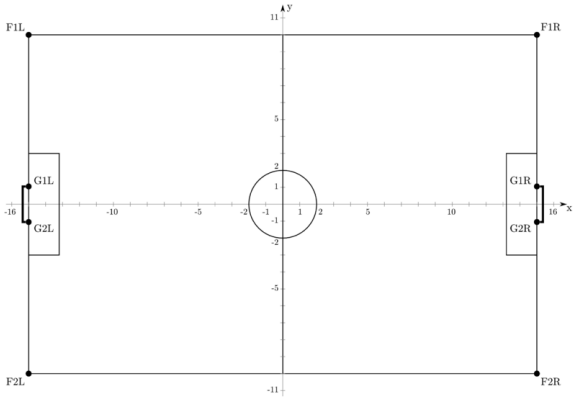
Simulované hřiště má rozměry 30 na 20 metrů. V porovnání se skutečným je tedy o něco menší. Branky jsou 2,1 m dlouhé a 0,8 m vysoké. Fotbalový míč má poloměr 4 cm a váží 26 g. Centrální kruh má poloměr 2 m. Visuálně je pro jednoduchost robotům reprezentován jako desetiúhelník. Kolem hřiště se nachází hranice, která má 10 m v každém směru, za tuto hranici se agenti nemají možnost dostat.

[4] http://www.robocup.org
[5] http://wiki.robocup.org/Soccer_Simulation_League
[6] http://www.ode.org/

Obrázek 2.1: Ukázka simulačního prostředí SimSpark zobrazeného přes defaultní monitorovací aplikaci.

Zdroj

Obrázek 2.2: Souřadnicový systém hřiště



Zdroj

Každý hráč je založen na modelu Aldebaran Nao 3. Na výšku měří 57 cm a váží 4,5 kg. Robot má celkem 22 pohyblivých kloubů: 6 v každé noze, 4 v každé ruce a 2 v oblasti krku. Aby mohl agent vnímat své prostředí a reagovat na něj je vybaven senzory a motory. Senzory předávají v každém simulačním cyklu přesnou informaci o úhlech, které svírají všechny klouby. Motory zase umožňují nastavit rychlost změny úhlu a jeho konečnou hodnotu. Zorné pole agenta má 120 stupňů. Obrazová informace o prostředí nacházejícím se v zorném poli je agentovi předávána každý 3. simulační cyklus v podobě vzdálenosti objektů a úhlu v jakém jsou pozorovány. Tyto informace ovšem nejsem úplně přesné, agentovi jsou předávány lehce zkreslené. Pro vzájemnou komunikaci může agent každých 40 ms vyslat zprávu o maximální délce 20 bytů.

Zápas proti sobě hrají dva týmy každý s 11 hráči. Každý zápas trvý 600 sekund a je rozdělen na dva poločasy.

2.2 Analýza agenta UT Austin Villa

Jedním z nejúspěšnějších týmů robotické fotbalu je tým Austinské Univezity v USA, UT Austin Villa. Tento tým zveřejňuje mnoho materiálů a prací, v nichž popisují, jakými způsoby svůj tým vylepšují. V této části budou rozebrána některá vylepšení strategického uvažování jejich agenta, které představili pro ročník 2015.

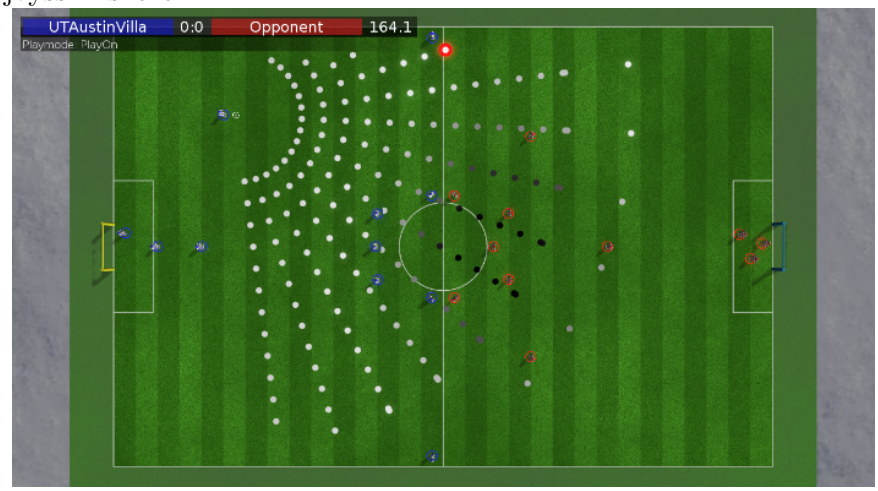
2.2.1 Kop s proměnlivou délkou

V ročníku 2014 ovládal agent pouze 4 techniky kopání na míče.

1. Přihrávka na maximální vzdálenost 5 metrů
2. Kop po zemi do vzdálenosti kolem 15 metrů
3. Kop vzduchem do vzdálenosti kolem 15 metrů
4. Dlouhý kop až na vzdálenost 20 metrů

Toto rozdělení však neumožňovalo přesnější nahrávání volným hráčům a tak se tvůrci rozhodli přidat 13 nových kopů tak, aby byl jejich agent schopný přihrávat na vzdálenost 3 do 15 metrů s přesností na jeden metr. Díky této granularitě umístění míče může tým UT Austin Villa vyhodnotit každou z jeho potencionálních nových pozic a kopnout jej do oblasti s nejvyšší hodnotou. Skóre každé oblasti je počítáno pomocí statické funkce, která bere v potaz vzdálenost nové pozice od soupeřovy brány, počet soupořových hráču v okolí a také vzdálenost nejbližšího spoluhráče od nové pozice míče.

Obrázek 2.3: Možnosti kopnutí míče, čím světlejší tím větší skóre. Červeně zvýrazněná oblast s nejvyšším skóre



Zdroj

2.2.2 Rozehrání ze středu

Pro ročník 2015 bylo představeno nové pravidlo, které zakazuje vstřelit gól ihned po rozehrání ze středového kruhu. Jelikož tuto strategii tým AU Austin Villa často používal byli nuceni chování svých agentů upravit. Vytvořili a naprogramovali proto agentům následují

novou strategii zahájení hry. Ihned po rozehrání se po pravém i levém křídle rozběhne hráč směrem k soupeřově bráně. Hráč uprostřed kruhu buď nahraje míč dozadu nebo jej rovnou přihraje jednomu z hráčů na křídle. Zadní hráč který se jako první dostane k míči pak následně pomocí techniky kopu s proměnlivou délkou přihradu tomu z hráčů na křídle, který je je výhodnější pozici. Po obdržení nahrávky pak už jen křídlový hráč kope přímo na soupeřovu branku.

Obrázek 2.4: Strategie rozehrání nové hry. Vlevo přímá nahrávka na křídlo, vpravo s nahrávkou dozadu

Zdroj

2.2.3 Zhodnocení

Tým UT Austin Villa využívá ve svém agentovi chování, které je velice precizně definováno. Strategické rozhodování agenta je řízeno pomocí manuálně naprogramovaných pravidel, která skvěle fungují. Ve svém agentovi bych ale chtěl dosáhnout toho, aby se všechny složitější vzorce chování naučil sám pouze s využitím základních akcí na základě odehrání velkého počtu her.

2.3 Základní fotbalový agent

Poskytnutý základní agent má implementovánu veškerou základní funkcionalitu včetně komunikace se serverem. Všechna data přicházející ze simulačního serveru jsou nejprve zparsována pomocí funkcí v souboru `agent/parser/parser.cc`. Zpracovaná data jsou předána metodě `NaoBehavior::Think()`, která z nich vyextrahuje všechny důležité informace, naplní jimi objekt `worldModel` a zavolá metodu `NaoBehavior::Act()`, která obstarává vykonávání akcí. Jaká akce se bude vykonávat vrací metoda `NaoBehavior::selectSkill()`. V této metodě bude později implementováno rozhodování pomocí neuronové sítě.

Na základě pozorování a informací od spoluhráčů je agent schopný určit svou pozici na hřišti. Základní agent obsahuje metody pro převod mezi relativními (vztažené k danému agentovi) a absolutními (vztažené k samotnému hřišti) souřadnicemi. Tyto informace jsou dostupné v objektu `worldModel`. Tento objekt poskytuje například následující informace:

- `worldModel->getMyPosition()` - informace o poloze hráče
- `worldModel->getMyAngDeg()` - slouží k získání orientace hráče
- `worldModel->getBall()` - vrací pozici míče

Veškeré dovednosti, které je daný agent schopen vykonat jsou definovány ve složce skills. Každá dovednost je sekvence určitých úkonů, které jsou popsány pomocí textových instrukcí určujících které klouby se mají ohýbat, kdy, v jakém úhlu a jakou rychlostí.

Pro pohyb na hřišti ovládá agent následující akce `NaoBehavior::goToTargetRelative` a `NaoBehavior::goToTarget`. Argumentem těchto funkcí je objekt `VecPosition`.

Základní agent je vybanen třemi technikami pro práci s míčem - driblováním, krátkou střelou a dlouhou střelou.

Složitějšího chování agenta bylo odstraněno včetně algoritmů určujících jaká akce se má vykonat. Toto je oblast, kterou bych chtěl pomocí posilovaného učení zlepšovat.

2.4 DeepCL

Jedná se o OpenCL[7] knihovnu určenou na vytváření a trénování konvolučních neuronových sítí pro zpracovávání obrazů.

2.4.1 Charakteristika použitých vrstev

Neuronová síť, kterou byla použita při všech experimentech má následující konfiguraci.

```
EasyCL *cl = new EasyCL();
NeuralNet *net = new NeuralNet( cl );

net->addLayer( InputLayerMaker::instance()->numPlanes(planes)->imageSize(size) );
net->addLayer( ConvolutionalMaker::instance()->filterSize(5)->numFilters(8)
                                             ->biased()->padZeros() );
net->addLayer( ActivationMaker::instance()->relu() );
net->addLayer( ConvolutionalMaker::instance()->filterSize(5)->numFilters(8)
                                             ->biased()->padZeros() );
net->addLayer( ActivationMaker::instance()->relu() );
net->addLayer( FullyConnectedMaker::instance()->imageSize(1)->numPlanes(100)
                                              ->biased() );
net->addLayer( ActivationMaker::instance()->tanh() );
net->addLayer( FullyConnectedMaker::instance()->imageSize(1)
  ->numPlanes(numActions)->biased() );
net->addLayer( SquareLossMaker::instance() );
```

Vstupní vrstva

Každá konvoluční síť tvořená pomocí knihovny DeepCL musí mít právě jednu vstupní vrstvu. Parametry vstupní vrstvy je počet rovin zpracovávaného obrazu a jeho velikost. Tato vrstva se přidává následujícím příkazem.

```
net->addLayer( InputMaker::instance()->numPlanes(numPanes)->imageSize(imageSize) )}
```

Aktivační vrstva

Aktivační vrstva je pomocnou vrstvou knihovny DeepCL, tato vrstva udává, jaká aktivační funkce bude použita na výstup předchozí vrstvy. DeepCL podporuje následující typy aktivačních vrstev resp. funkcí.

- linear()

[7] https://cs.wikipedia.org/wiki/OpenCL

- relu()
- elu()
- sigmoid()
- tanh()
- scaledtanh() // 1.7159 * tanh(0.66667 * x)

Chybí funkce jednotkové skoku, která se v konvolučních sítích nepoužívá. Aktivační vrstvě musí předcházet jiná vrstva a její použití se zapisuje následovně.

```
net->addLayer( ActivationMaker::instance()->relu() );
```

Konvoluční vrstva

Konvoluční vrstva je základním stavebním prvkem konvolučních neuronových sítí. Pomocí parametrů lze nastavit počet a velikost filtrů i aktivační funkci aplikovanou na výstup.

```
net->addLayer( ConvolutionalMaker::instance()->numFilters(32)->filterSize(5)
                                              ->relu()->biased() );
```

Plně propojená vrstva

Jedná se o vrstvu známou z tradičních neuronových sítí, tvoří ji neurony, které mají na vstupu hodnoty všech výstupů předchozí vrstvy. Úkolem této vrstvy je klasifikace příznaků získaných v předchozí vrstvě do skupin.

```
net->addLayer( FullyConnectedMaker::instance()->numPlanes(2)->imageSize(28) );
```

Výstupní vrstva

Každá neuronová síť vytvořená pomocí knihovny DeepCL musí mít právě jednu výstupní vrstvu, jedná se o pomocnou vrstvu, která pouze udává jaká výstupní funkce se použije na výstup předcházející vrstvy. Jako výstupní vrstvu lze použít jednu z následujích třech funkcí.

```
net->addLayer( SquareLossMaker::instance() );
net->addLayer( CrossEntropyMaker::instance() );
net->addLayer( SoftMaxLayer::instance() );
```

2.4.2 Učební scénáře

Základem pro Q-learning v této knihovně je učební třída učebního scénáře `Scenario`. Objekty této třídy musí implementovat funkce popisující zadanou úlohu.

```
class Scenario {
public:
    virtual ~Scenario() {}
    virtual void print() {} // optional
    virtual void printQRepresentation(NeuralNet *net) {} // optional
    virtual int getPerceptionSize() = 0;
```

```
    virtual int getPerceptionPlanes() = 0;
    virtual void getPerception(float *perception) = 0;
    virtual void reset() = 0;
    virtual int getNumActions() = 0;
    virtual float act(int index) = 0;
    virtual bool hasFinished() = 0;
};
```

- Funkce `getPerceptionSize` vrací velikost vstupního obrazu.
- Funkce `getPerceptionPlanes` vrací počet rovin vstupního obrazu.
- Funkci `getPerception` je jako argument předáno prázdné pole hodnot typu `float`. Velikost pole odpovídá počtu pixelů vstupního obrázku vynásobené počtem jeho rovin. Úkolem této funkce je zapsat do tohoto pole aktuální vizuální vjem agenta.
- Funkci `reset` se volá před zahájením nové etapy a slouží k inicializaci prostředí.
- Funkce `getNumActions` vrací počet dostupných akcí agenta.
- Funkci `act` je jako argument předán index akce, která byla pro současný stav vybrána jako nejlepší. Návratovou hodnotou typu `float` této funkce je výše odměny pro agenta za vykonání této akce. Tato funkce určuje jaký index akce má jaký význam například 0 jako pohyb doprava, 1 jako pohyb doleva atd. Implementace jednoduché funkce act může vypadat následovně.

```
int dx = 0;
int dy = 0;
switch(actionIndex) {
    case 0:
        dx = 1;
        break;
    case 1:
        dx = -1;
        break;
    case 2:
        dy = 1;
        break;
    case 3:
        dy = -1;
        break;
}
int newX = posX + dx;
int newY = posY + dy;

if(newX == goalX && newY == goalY) {
    finished = true;
    posX = newX;
    posY = newY;
    return 1;
```

```
} else {
    posX = newX;
    posY = newY;
    return -0.1f;
}
```

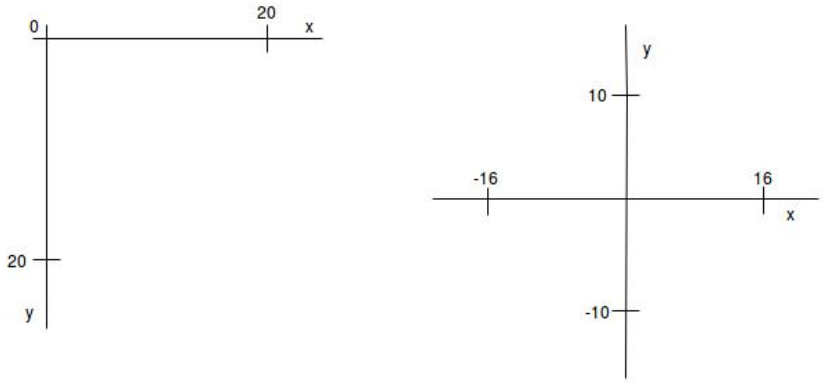
- Funkce `hasFinished` pokud tato funkce vrátí true je etapa ukončena, zavolá se funkce reset a odstartuje se nová etapa.
- Nepovinné funkce `print`, `printQRepresentation` lze použít pro vizuální validaci učebního scénáře.

Takto vytvořený scénář se následně předává jako parametr kontruktoru objektu třídy `QLearner`. Ve výchozím nastavení se používá tento objekt pro upravování vah historii posledních 32 akcí. Toto lze upravit změnou atributu `maxSamples` tohoto objektu. Dalšími atributy tohoto objektu, které lze upravit jsou Discount Factor - `lambda` a `epsilon` udávající pravděpodobnost objevování místo využívání. Po zavolání metody `run` je zahájeno učení. Implementace této metody je nekonečný cyklus, který nemá žádnou ukončovací podmínku. O ukončení procesu učení se tak musí starat objekt `Scenario`. V mém případě se v metodě `reset` tohoto objektu pravidelně kontroluje kolik tahů zabralo ukončení X posledních etap. Pokud je tato suma menší než nastavený práh je učební seance ukončena a váhy neuronové sítě jsou vyexportovány do binárního souboru `weights`.

2.5 Provázání DeepCL a agenta

Propojení fotbalového agenta a neuronové sítě nebylo až tak složité, jednou z největších výzev byl pouze převod mezi souřadnicovými systémy a mapování dotupných akcí na akce robotického hráče. Zatímco neuronová síť používá pro osu X i Y souřadnice z intervalu $< 0, 20 >$, fotbalový agent používá souřadnice osy X z intervalu $< -16, 16 >$ a $< -11, 11 >$ pro osu Y ovšem v opačném směru než neuronová síť jak je patrné z obrázku **??**.

Obrázek 2.5: Vlevo souřadnicový systém používaný neuronovou sítí, vpravo souřadnicový systém simulátoru



Provázání funguje následovně, při inicializaci agenta je inicializována i neuronová síť a její parametry jsou nastaveny pomocí hodnot v souboru `weight`. Následně v hráčově metodě `NaoBehavior::selectSkill()` jsou získány souřadnice bodů zájmu. Tyto souřadnice

jsou převedeny do souřadnicového systému neuronové sítě. Následně je s využitím těchto souřadnic naplněno pole perception a předáno neuronové síti. Nejvhodnější akce se vybírá následující algoritmem.

```
net->forward(perception);
float highestQ = 0;
int bestAction = 0;
float const*output = net->getOutput();
for(int i = 0; i < 4; i++) {
        if(i == 0  output[i] > highestQ) {
                highestQ = output[i];
                bestAction = i;
        }
}
```

Následně už je potřeba pouze namapovat akce použité v učební scénářem na akce agenta, například akci pohyb doleva lze implementovat takto.

```
newPosition = VecPosition(getSoccerPositionX(posX - 1, size), getSoccerPo
return goToTarget(newPosition);
```

Kapitola 3

Provedené experimenty

Všechny zdrojové kódy provedených experimentů spolu s výsledným snímkem obrazovky jsou uloženy ve složce `experiments`. V experimentech, které jsem prováděl jsem se rozhodl neuvyužít některé z pokročilejších funkcí agenta. Experimenty jsem stavěl pouze na základních příkazech, protože si myslím, že jen tak se naplno projeví síla q learningu.

Veškeré nápady jsou nejdříve ověřeny v režimu simulace, simulační režim nepracuje s rcsserverem. Využívají velice zjednodušené prostředí, naimplementované v jazyce C.

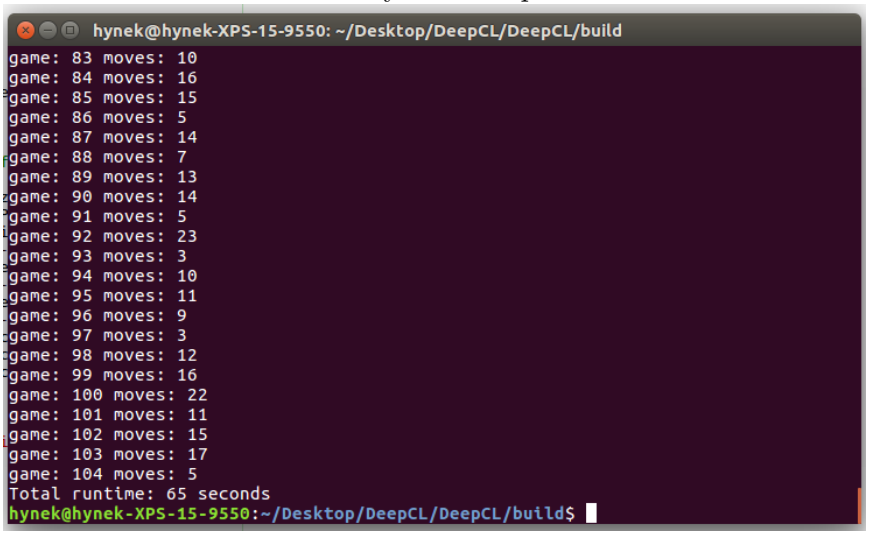
Ve většině experimentů byla použita velikost prostoru 20x20, plocha hřiště je tedy rozdělena na síť ve které má každý bod velikost 1,5 x 1 metr. Pouze při validaci nových konceptů byl prostor zmenšen tak aby se platnost myšlenky prokázala rychleji.

3.0.1 Experiment 0 - dosažení středu hřiště

Experiment 0 vyhází z ukázkové implementace Q-learningu v knihovně DeepCL a pouze mírně ji upravuje. Cílem experimentu je naučit agenta dosáhnout středu hřiště z kteréhokoliv bodu na hřišti. V tomto experimentu agent obdrží pozitivní odměnu pouze po dosažení středu. Naopak za každou provedenou akci je mu udělena malá negativní odměna a negativně je také hodnocen pokus o pohyb mimo hranice hřiště.

- Velikost hřiště: 20x20 bodů
- Akce agenta: pohyb 1 bod doleva, pohyb 1 bod doprava, pohyb 1 bod dolů, pohyb 1 bod nahoru
- Reward funkce: Za každou provedenou akci udělena odměna -0.1, za pohyb mimo hranici hřiště odměna -0.5.
- Ukončovací podmínka etapy: Agent provede více než 500 akcí, udělena odměna -1 a etapa je ukončena. Agent dorazí do středu hřiště, udělena odměna 1 a etapa ukončena
- Ukončovací podmínka experimentu: Agent v 50 po sobě jdoucích etapách dorazí do středu hřiště s využitím méně než 25 pohybů.
- Výsledek experimentu (obrázek 3.1): Experiment úspěšně ukončen po odehrání 104 etap. Celkový čas experimentu 64 s.

Obrázek 3.1: Výsledek Experimentu 0

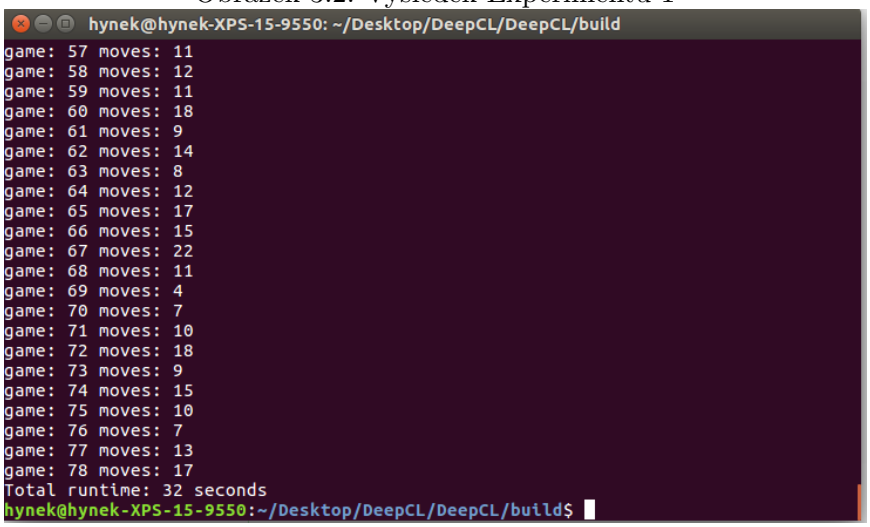


3.0.2 Experiment 1 - dosažení středu hřiště, konkrétnější reward funkce

Experimentu 1 výchází z předchozího pokusu. Cílem bylo vyzkoušet upravenou reward funkci, která agentovi poskytuje více zpětné vazby o správnosti jeho počínání a dopad této změny na rychlost učení. Agent dostává malou pozitivní odměnu pokaždé když vykoná pohyb ke středu hřiště.

- Velikost cvičiště: 20x20 bodů
- Akce agenta: stejné jako v Experimentu 0
- Reward funkce: Za každý provedený pohyb, který agenta přiblížil ke středu udělena odměna 0.1, za pohyb mimo hranici hřiště odměna -0.5, za jakýkoliv jiný pohyb odměna -0.1
- Ukončovací podmínka etapy: stejné jako v Experimentu 0
- Ukončovací podmínka experimentu: stejné jako v Experimentu 0
- Výsledek experimentu (obrázek 3.1): Experiment úspěšně ukočen po odehrání 78 etap. Celkový čas experimentu 32 sekund. Podařilo se dokázat, že reward funkce poskytující průběžnou pozitivní zpětnou vazvu dokáže výrazně urychlit proces učení.

Obrázek 3.2: Výsledek Experimentu 1

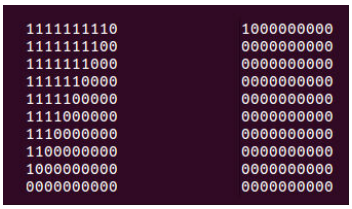


3.0.3 Experiment 2 - dosažení středu hřiště, rotace

Expriment 2 simuluje složitější chování agenta. Místo 4 akcí pohybu ve všech směrech má agent v tomto experimentu pouze 3 akce a to otočení o 45 stupňů v obou směrech a chůzi kupředu. Celkem teda exituje až 8 možností agentovy orientace. Pro rychlejší ověření těchto domněnek jsem v tomto experimentu zmenšil velikost hřiště na 10x10 bodů.

Tento experiment je také zejména tím, že agent nejprve musí vykonat několik akcí rotace, za kterou dostává negativní odměnu a až poté akci chůze. Dokazuje se tím jedna ze silných stránek Q-learningu, agent je schopný si osvojit i takové chování, které se krátkodobě zdá nevýhodné avšak vede k velkému zisku v dlouhodobějším měřítku. Další výzvou tohoto experimentu byla reprezentace agentovi orientace na hřišti. Z tohoto důvodu je experiment rozdělen na dvě části. V každé části je pro informaci o úhlu použitá jiná grafická forma jak je znázorněno na obrázku 3.3. Vstupní obraz, který doposud obsahoval jen jednu rovinu s vyznačenou pozicí agenta je tak rozšířen o druhou rovinu, která kóduje agentovu orientaci.

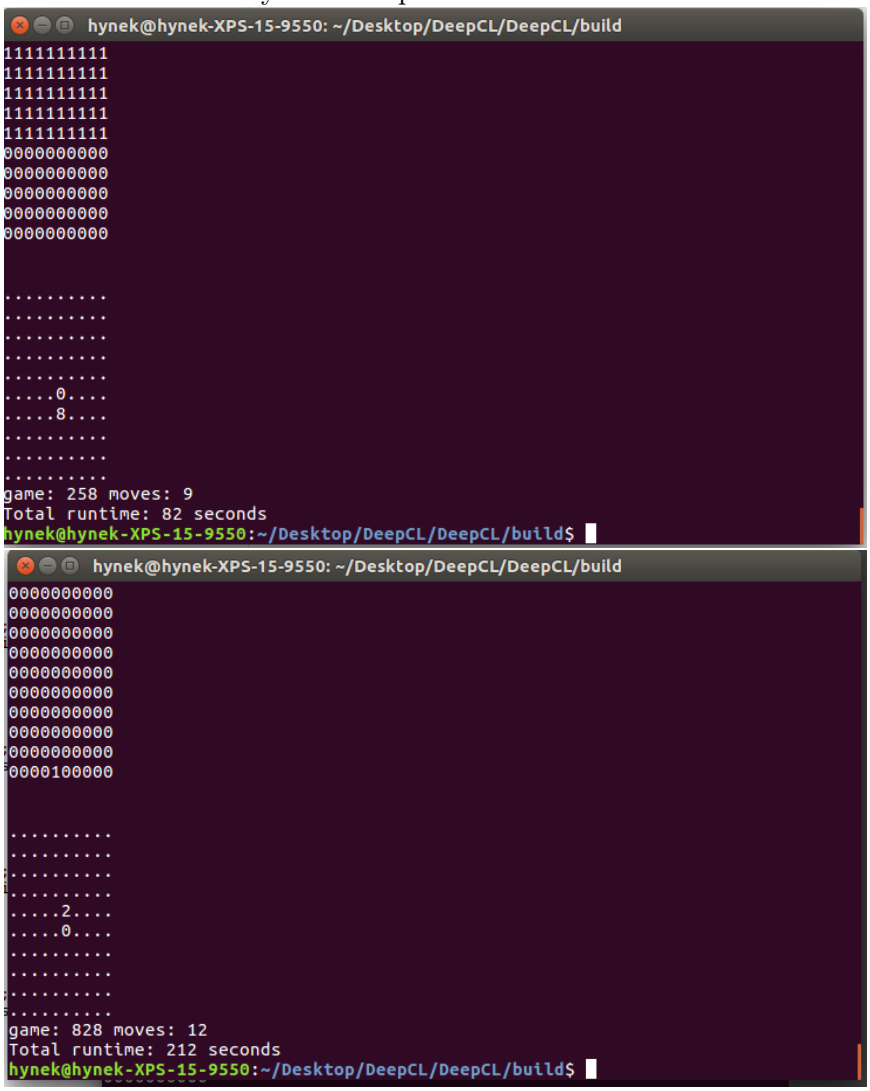
Obrázek 3.3: Dvě možnosti reprezentace stejné orientace. Vlevo výraznějším příznak, vybarvená je celá odpovádající polovina orientační roviny, vpravo nevýrazný příznak v podobě pouze jednoho vybarveného bodu.



Na tento experiment bych také rád později navázal přidáním akce kop přímo před sebe. Toto nastavení by již mělo být dostatečně flexibilní na to, aby se i při následném přenosu na platformu SimSpark objevili zajímavější vzorce agentova chování.

- Velikost cvičiště: 10x10 bodů
- Akce agenta: pohyb o 1 bod a to i v diagonálním směru, otočení na místě o 45 stupňů v obou směrech
- Reward funkce: Za každý provedený pohyb, který agenta přiblížil ke středu udělena odměna 0.1, za pohyb mimo hranici hřiště odměna -0.5, za jakýkoliv jiný pohyb odměna -0.1
- Ukončovací podmínka etapy: stejné jako v Experimentu 0
- Ukončovací podmínka experimentu: stejné jako v Experimentu 0
- Výsledek experimentu A (obrázek 3.4): Experiment úspěšně ukočen po odehrání 258 etap. Celkový čas experimentu 82 sekund.
- Výsledek experimentu B (obrázek 3.4): Experiment úspěšně ukočen po odehrání 828 etap. Celkový čas experimentu 212 sekund.
- Výsledek experimentu: Ukázalo se, že po zakódování informace o agentově orientace je mnohem výhodnější použití výraznějšího příznaku použíté v Experimentu 2A. Tento experiment také ukázal, že agent je schopen si osvojit i složitější vzorce chování ve který musí vykonat i několik negativní akcí pro získání budoucí odměny.

Obrázek 3.4: Výsledek Experimentu 2A nahoře a 2B dole



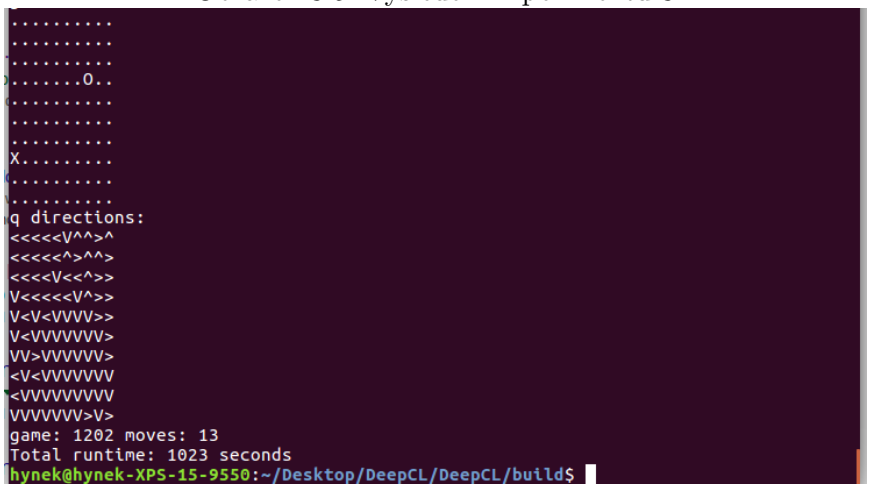
3.0.4 Experiment 3 - nalezení cíle

V Experimentu 3 jsem se vrátil zpět k základním akcím pohybu ve všech směrech. V tomto experimentu je pozice cílového pole, kam má agent dojít zvolena náhodně pro každou etapu. Tímto je simulována situace kdy agent zná pozici míče na hřišti a chce k němu dojít. Informace o poloze cíle je agentovi předána v nové vstupní rovině. V této rovině má cílový bod hodnotu 1 a všechny ostatní body jsou nulové.

- Velikost cvičiště: 10x10 bodů
- Akce agenta: stejné jako v Experimentu 0
- Reward funkce: Za každý provedený pohyb, který agenta přiblížil k cíli udělena odměna 0.1, za pohyb mimo hranici hřiště odměna -0.5, za jakýkoliv jiný pohyb odměna -0.1

- Ukončovací podmínka etapy: Agent provede více než 1000 akcí, udělena odměna -1 a etapa je ukončena. Agent dorazí na určené místo na hřišti, udělena odměna 1 a etapa ukončena
- Ukončovací podmínka experimentu: Agent v 50 po sobě jdoucích etapách dorazí na určené místo s využitím méně než 50 pohybů.
- Výsledek experimentu (obrázek 3.5): Experiment úspěšně ukočen po odehrání 1202 etap. Celkový čas experimentu 1023 sekund. Ačkoliv experiment probíhal na zmenšeném hřišti ukázalo se, že naučení požadovaného chování je opravdu náročné. Malá rychlost učení je zřejmě způsobena tím, že příznak udávající pozice míče na hřišti je velmi malý, jedná se o pouze jeden pixel.

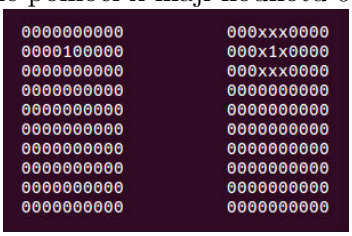
Obrázek 3.5: Výsledek Experimentu 3



3.0.5 Experiment 4 - nalezení cíle, cíl označen výrazněji

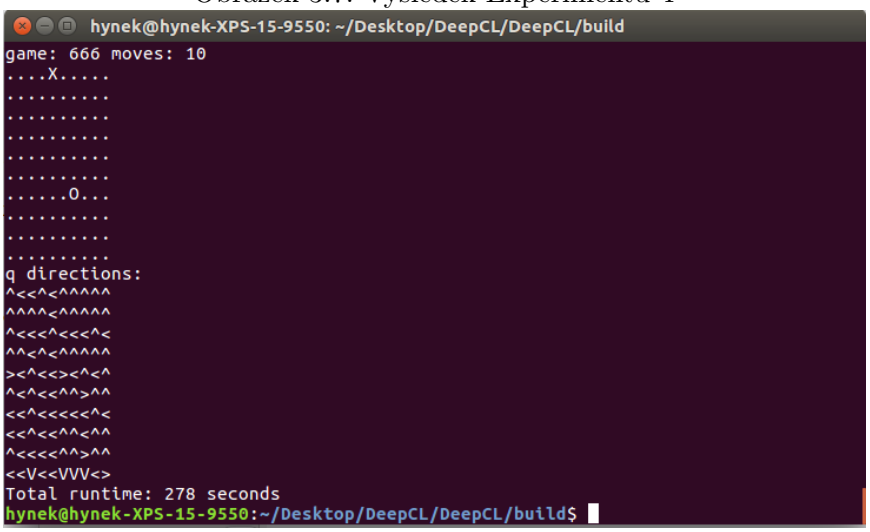
Cílem tohoto experimentu je ověřit, že použitá konvoluční síť lépe funguje s většími příznaky. V tomto experimentu je okolí cíle vyplněno body s hodnotou 0.5. Rozdíl těchto reprezentací je znázorněn na obrázku 3.6. Pokud je má domněnka správná, v tomto experimentu by měl učící proces probíhat o mnoho rychleji.

Obrázek 3.6: Dvě možnosti reprezentace pozice cíle. Vlevo cíl reprezentován pouze jedním bodem, vpravo je body označené pomocí x mají hodnotu 0.5



- Velikost cvičiště: 10x10 bodů
- Akce agenta: stejné jako v Experimentu 0
- Reward funkce: stejná jako v Experimentu 3
- Ukončovací podmínka etapy: stejná jako v Experimentu 3
- Výsledek experimentu (obrázek 3.8): Experiment úspěšně ukočen po odehrání 666 etap. Celkový čas experimentu 278 sekund. Pomocí tohoto experimentu se podařilo prokázat, že pro zpracování pomocí použité konvoluční sítě je o mnoho lepší použití výraznějších příznaků.

Obrázek 3.7: Výsledek Experimentu 4



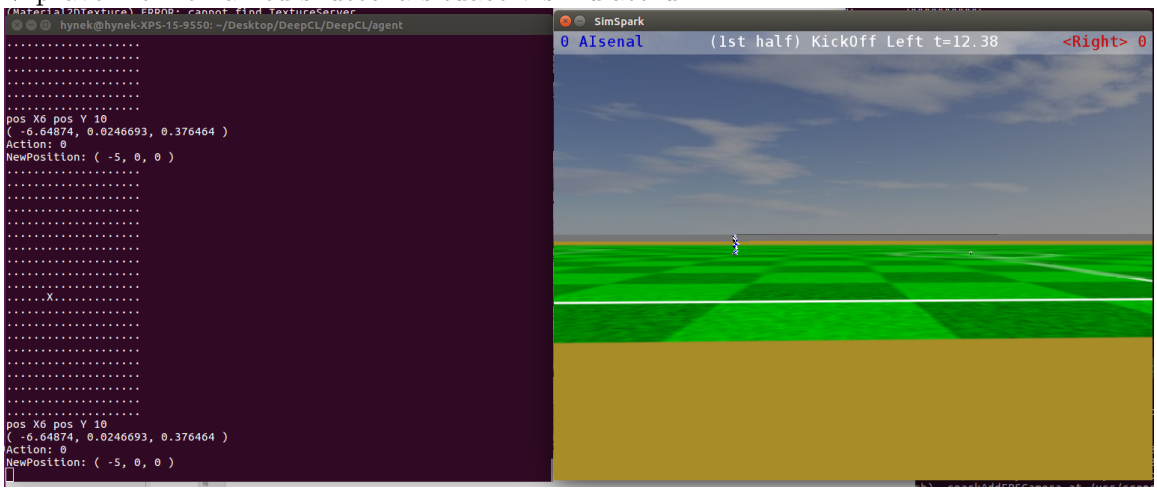
3.0.6 Experiment 5 - spojení fotbalového agenta a naučené neuronové sítě

Toto je první experiment ve kterém dojde k propojení fotbalového agenta s neuronovou sítí. Jako základ byla použita natrénovaná síť z Experimentu 1. Dále bylo nutné implementovat v agentovi stejný set akcí jako byl dostupný v Experimentu 1. K tomuto účelu jsem využil agentovu dovednost `goToTarget`. Této dovednosti stačí předat absolutní souřadnice na které má agent dojít.

- Velikost cvičiště: Reálné hřiště
- Akce agenta: stejné jako v Experimentu 0
- Reward funkce: -
- Ukončovací podmínka etapy: -
- Výsledek experimentu (obrázek **??**): Úspěšně se podařilo použít natrénovanou neuronovou síť pro ovládání agenta robotického fotbalu. Agent dokázal z libovolné pozice na hřišti dorazit středu hřiště.

Složka s tímto experimentem obsahuje soubor s uloženými vahami a také soubor strategy.cc použitý pro tento experiment.

Obrázek 3.8: Průběh Experimentu 5. V levém okně debugovací výstup agenta na kterém jde vidět vnitřní reprezentaci situace na hřišti a také parametry předané funkci GoToTarget. V pravém okně náhled skutečná situace v simulátoru.



Závěr

Tato práce byla zaměřená na problematiku implementace agenta založeného na posilovaném učení, který bude ovládat hráče robotického fotbalu v prostředí simulátoru SimSpark. Tohoto agenta se mi úspěšně podařilo implementovat s využitím metody Q-learning.

Práce na tomto projektu mi dala velkou lekci v oblasti plánování vývoje a ukázala mi hlavní sílu agilních vývojových metodik. Stalo se, že jsem strávil několik týdnů vývojem jedné části řešení, konkrétně automatizační knihovny pro spouštění velkého počtu simulací, a následně jsem zjistil, že pro řešení tohoto problému bude nepoužitelná. Příště budu při řešení podobně rozsáhlého projekt rozhodně postupovat agilně. Mít co nejdříve klidně nedokonalý funkční protyp řešení a ten následně vylepšovat, než se zaměřit na jednu část projektu a po čase zjistit, že je nepoužitelná pro zbytek programu.

Bohužel jsem nemohl porovnat výkon mého agenta se soutěžními agenty, jelikož nemá implementovány akce pro práci s míčem a tudíž nemůže dát gól ani rozehrát hru.

Práce poskytuje solidní základ na který lze navázat v mnoha různých směrech. Jedním z nich je provázání učebních scénářů se simulačním serverem tak, aby se agent mohl učit v simulačním prostředí z her odehraných proti reálným soupeřům. Dále se nabízí, jelikož je v týmu 11 hráču, implementace sdílení jedné neuronové sítě jejíž váhy upravují všichni agenti. Učební přínos jedné odehrané hry by tak mohl být až 11x větší.

Jako další rozšíření by bylo možné přidat akce pro práci s míčem a upravit reward funkci tak, aby odměňovala zmenšování vzdálenosti od soupeřovy branky. Složitější vzorce chování možná budou vyžadovat úpravu používané neuronové sítě.

Literatura

[1] *An Intuitive Explanation of Convolutional Neural Networks*. [Online; navštíveno 12.05.2017].
URL https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[2] *IBM100 - Deep Blue*. [Online; navštíveno 12.05.2017].
URL http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/

[3] *Markovův rozhodovací proces*. [Online; navštíveno 12.05.2017].
URL https://cs.wikipedia.org/wiki/Markov%C5%AFv_rozhodovac%C3%AD_proces

[4] *New record in ImageNet Challenge set by Microsoft Researchers*. [Online; navštíveno 12.05.2017].
URL http://news.thewindowsclub.com/imagenet-challenge-microsoft-research-71373/

[5] *Reinforcement Learning*. [Online; navštíveno 12.05.2017].
URL http://www.cse.unsw.edu.au/~cs9417ml/RL1/algorithms.html

[6] *Strojové učení a MATLAB*. [Online; navštíveno 12.05.2017].
URL http://sciencemag.cz/strojove-uceni-a-matlab/

[7] *Turing test*. [Online; navštíveno 12.05.2017].
URL https://en.wikipedia.org/wiki/Turing_test

[8] *Umělá inteligence - význam - IT Slovník*. [Online; navštíveno 12.05.2017].
URL https://it-slovnik.cz/pojem/umela-inteligence

[9] KRAJČOVIČOVÁ, M.: *Konvoluční neuronová síť pro zpracování obrazu*. Diplomová práce.
URL https://dspace.vutbr.cz/xmlui/handle/11012/40018

[10] Littman, M. L.: *Markov Games as a Framework for Multi-Agent Reinforcement Learning. Proceedings of the 11th International Conference on Machine Learning (ICML)*, 1994: str. 157–163, doi:10.1016/B978-1-55860-335-6.50027-1, article.
URL https://www.researchgate.net/publication/2799903_Markov_Games_as_a_Framework_for_Multi-Agent_Reinforcement_Learning

[11] McCulloch, W. . P.: *Bulletin of Mathematical Biophysics*. 1943, 5-115 s., doi:10.1007/BF02478259.

[12] Silver, D.; Huang, A.; Maddison, C. J.; aj.: Mastering the game of Go with deep neural networks and tree search. *Nature*, ročník 529, č. 7587, Jan 2016: s. 484–489, ISSN 0028-0836, article.
URL http://dx.doi.org/10.1038/nature16961

[13] Sutton, R. S.; Bartos, A. G.: *Reinforcement Learning*. 1998, ISBN 9780262193986.

Seznam obrázků