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

NEURONOVÉ SÍTĚ PRO HRU GOMOKU

PLAYING GOMOKU WITH NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATÚŠ BAKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20168/2016/xbakom01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Bako Matúš**
Obor: Informační technologie
Téma: **Neuronové sítě pro hru gomoku**
Playing Gomoku with Neural Networks
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy konvolučních sítí, zpětnovazebního učení a vyhledávacích algoritmů pro hraní her.
2. Vytvořte si přehled o současném využití konvolučních sítí v pro hraní her.
3. Navrhněte konkrétní metodu hraní hry gomoku.
4. Připravte napojení na některé z používaných gomoku turnajových rozhraní.
5. Implementujte navrženou metodu a otestujte ji proti existujícím gomoku AI.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte krátké video prezentující vaši práci, její cíle a výsledky.

Literatura:

- D. Silver et al.: Mastering the game of Go with Deep Neural Networks & Tree Search. Nature 2016.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4.

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: **Hradiš Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cieľom tejto bakalárkej práce je vytvoriť umelú inteligenciu, ktorá dokáže hrať hru Gomoku. Narozdiel od konvenčných metód prehľadávania stavového priestoru a ručně definovaných pravidiel kombinujem stochastické prehľadávanie s použitím konvolučných neurónových sietí. Strategická neurónová sieť určuje pravdepodobnostnú distribúciu pre všetky pozície na hracej ploche pri danej konfigurácii a ohodnocovacia sieť určuje pravdepodobnosť výhry. Natrénoval som architektúry neurónových sietí s rôznym počtom konvolučných vrstiev a s rôznymi veľkosťami konvolučných jadier. Z experimentov vyplinulo, že bez použitia ohodnocovacej siete a prehľadávacieho algoritmu je problematické zakončiť hru, avšak strategická sieť dokáže fungovať ako dostatočne silná heuristika pre výber ťahu. Napriek relatívne malej množine trénovacích dát je mnou vytvorená umelá inteligencia schopná poraziť nižšie umiestnené programy zo súťaže Gomocup.

Abstract

The goal of this thesis is to create an artificial intelligence for playing Gomoku. While conventional methods usually use state space search combined with predefined rules, this artificial intelligence uses state space search and learned neural networks. A strategic network computes probability distribution for given a board state and a value network determines outcome of the game from a given board state. I trained multiple architectures of neural networks with different number of convolutional layers and different sizes of convolution kernels. Experiments show, that it is problematic to end a game without using the value network or search algorithm, but the strategic network can be used as a heuristic for choosing next move. Despite using relatively small dataset, created artificial intelligence is capable of beating weaker programs from Gomocup competition.

Kľúčové slová

Umelá inteligencia, Neurónové siete, Gomoku

Keywords

Artificial intelligence, Neural networks, Gomoku

Citácia

BAKO, Matúš. *Neuronové sítě pro hru gomoku.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Hradiš Michal.

Neuronové sítě pro hru gomoku

Prehlásenie

Prehlasujem, že som túto prácu vypracoval samostatne, pod vedením Ing. Michala Hradiša, PhD. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Matúš Bako
17. mája 2017

Poďakovanie

Ďakujem Ing. Michalovi Hradišovi Ph.D. za pomoc a odborný dohľad pri písaní bakalárskej práce a implementácii riešenia. Taktiež ďakujem za prístup ku výpočtovým a úložným zariadeniam patriacim skupinám, ktoré prispievajú k Národnej infraštruktúrnej sieti MetaCentrum poskytovanej pod programom „Projekty veľkého výskumu, vývoja a inovácii infraštruktúr“(CESNET LM2015042).

Obsah

Kapitola 1

Úvod

Stolné hry sú niekoľko desaťročí terčom strojového učenia, pretože majú jasne definované pravidlá a kritériá výhry. Napriek tomu mnoho z nich poskytuje veľký počet dosiahnutelných konfigurácii, čo vytvára priestor pre súťaživosť a zdokonaľovanie. Pre hľadanie správneho ťahu sú používané rôzne metódy, napríkad prehľadávanie stavového priestoru, prípadne rôzne druhy učení. Líšia sa hlavne časovou a priestorovou zložitosťou.

V mojej práci sa snažím demonštrovať použitie konvolučných neurónových sietí pri hraní hry Gomoku. Konvolučné neurónové siete už boli použíté na vytvorenie umelej inteligencie pre hranie hier, medzi ktoré patria šach, Go, Backgammon a iné. Narozdiel od tradičných programov, ktoré majú natvrdo naprogramované isté pravidlá a heuristiky, používam na výber nasledujúceho ťahu neurónové siete kombinované s prehľadávaním stavového priestoru.

Experimenty boli vykonávané napojením umelej inteligencie na program Piskvork, ktorý umožnuje súťažiť proti iným programom, prípadne odohrané hry zaznamenávať. Strategická neurónová sieť, ktorá určuje pravdepodobnosti nasledujúceho ťahu pre všetky políčka, dokáže konkurovať programom Imro a Mentat bez prehľadávania stavového priestoru. Tieto programy boli v súťaži Gomocup umiestnené na 32. a 27. mieste. Z tohto vyplíva, že strategická sieť je vhodná ako heuristika pre výber nasledujúceho ťahu a môže byť použitá ako súčasť komplikovanejšieho algoritmu.

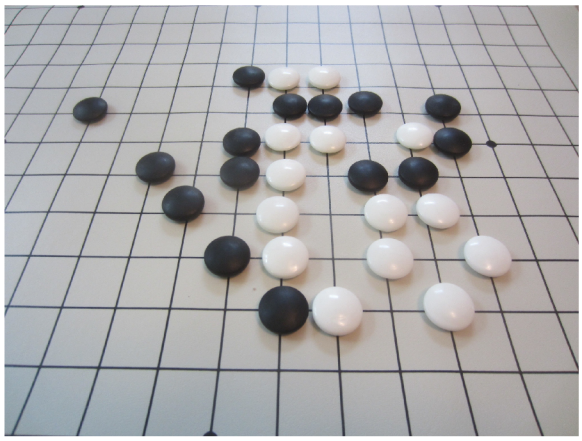
Kapitola 2

Stolné hry a piškvorky

Medzi najjednoduchšie hry pre viacerích hráčov patria stolné hry. Majú jednoduché pravidlá, presne stanovené kritérium výhry a ku hraní väčšinou stačí papier a pero, prípadne odlíšitelné žetóny. Platí však že pochopenie pravidiel je zväčša jednoduché, ale kým si človek povie, že danú hru naozaj vie hrať, musí odohrať veľké množstvo hier proti rôznym protihráčom, aby sa naučil rôzne výherné stratégie.

2.1 Piškvorky a Gomoku

Piškvorky je veľmi jednoduchá stolná hra s jasne definovanými pravidlami. Hráč vyhráva, ak položí dostatočný počet žetónov svojej farby v ortogonálnom alebo diagonálnom smere vedľa seba. Hracia doska pozostáva z troch riadkov a troch stĺpcov pre ukladanie žetónov. Je to hra s nulovým súčtom, čo znamená, že výťazný hráč vyhráva práve toľko, koľko jeho protihráč prehrá. Hrá sa iba o jeden bod a nie je možné, aby nastal nejaký prípad čiastkovej výhry. V prípade, že sa ani jednému z hráčov nepodarí položiť dostatočný počet žetónov jeho farby vedľa seba, nastáva remíza. V tom prípade sa ani jednému z hráčov nepripočíta bod a hrá sa odznova. Taktiež platí, že je to hra s úplnými informáciami, pretože obaja hráči



Obr. 2.1: Hracia plocha Gomoku pozostáva z rovnobežných a kolmých čiar, pričom kamene sa ukladajú na miesta, kde sa jednotlivé čiary pretínajú. Prevzaté z https://eightygames.wordpress.com/2013/05/17/gomoku-and-renju/.

vidia celú hraciu plochu, a žiadne informácie nie sú pre žiadneho hráča skryté. Piškvorky sú určené pre dvoch hráčov a nie sú založená na náhode.

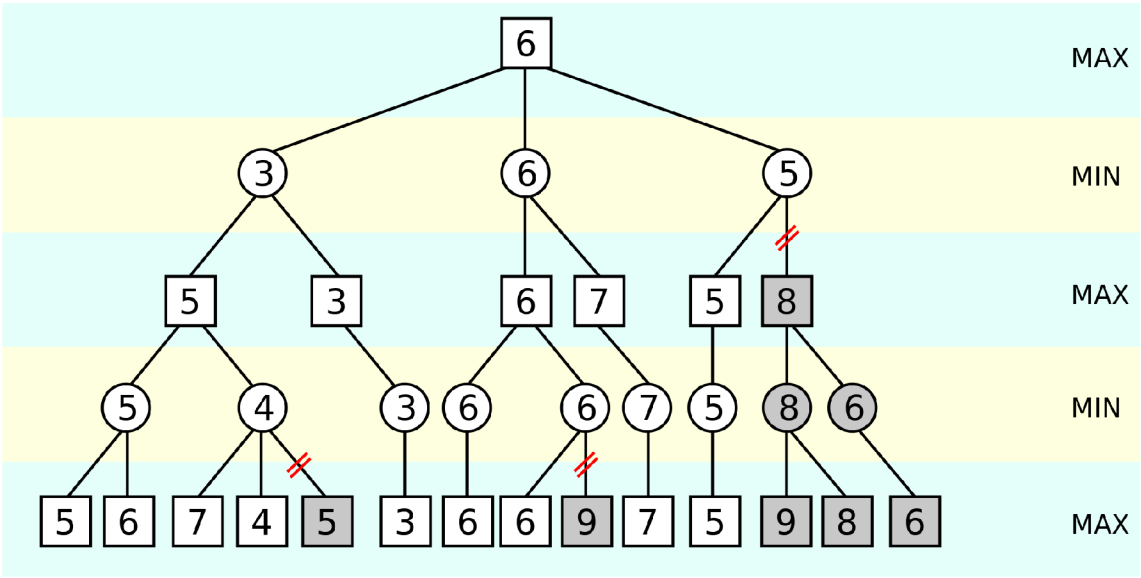
Nevýhodou známej verzie s hracou plochou o veľkosti 3x3 je malý stavový priestor. Preto som pre experimentovanie zvolil inú, ale zároveň veľmi podobnú, hru Gomoku. Jediný rozdiel v základných pravidlách je vo veľkosti hracej plochy. Niektoré varianty, ako napríklad Renju, počítajú s pätnástimi riadkami a stĺpcami, avšak ja som zvolil variantu, kde ich je devätnásť. Narozdiel od piškvoriek hráč vyhráva, ak má na hracej ploche v rade až päť kameňov jeho farby v ortogonálnom alebo diagonálnom smere.

2.2 Prístupy existujúcich algoritmov

V nasledujúcej podkapitole sú porovnané schopnosti štyroch algoritmov pre hranie Gomoku. Porovnanie pochádza z článku [1].

Ako prvý bol navrhnutý algoritmus *MiniMax*. Ten je bežne používaný pri hrách pre dvoch hráčov a hrách s nulovým súčtom. MiniMax funguje na princípe zostavenia stromu možných výsledkov. Obaja hráči sa snažia maximalizovať svoje skóre čím minimalizujú skóre druhého. Použitím rekurzie algoritmus pracuje tak, že pre každý možný ťah sa hľadá maximálne skóre hráča a minimálne skóre protihráča. Pretože gomoku má hraciu plochu o veľkosti 19x19, už iba na tretej úrovni zanorenia by sme počítali ohodnotenie pre $361^3 = 47\,045\,881$ ťahov. Takéto vetvenie spôsobuje náročnosť, kvôli ktorej nie je možné určiť nasledujúci ťah v rozumnom čase. Preto pôvodná implementácia tohto algoritmu pre daný problém nie je vhodná a algoritmus v pôvodnej podobe nebol použitý.

Prvý z algoritmov používa iba heuristickú funkciu pre určenie kvality ťahu. Heuristika sa pozerá na položené žetóny a počíta, koľko žetónov jednej farby je v rade. Tento počet určute ohodnotenie políčiek obklopujúcich dané žetóny v rade. Ohodnotenie je kladné alebo záporné podľa toho, či kamene v rade patria hráčovi alebo protihráčovi. Slabá stránka



Obr. 2.2: Ilustrácia Alfa-beta orezávania na konkrétnom príklade. V jednotlivých uzloch je vidieť striedavé vyhľadávanie minima a maxima v uzloch priamych potomkov. Prevzaté z https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning.

heuristiky je, že sa nepozerá dopredu v priebehu hry a nedokáže zistiť, či je hodnotenie pozícii relevantné vzhľadom aj na nasledujúce ťahy.

Ďalší algoritmus je tzv. Alfa-Beta orezávanie (*Alpha-Beta pruning*), ilustrovaný na obrázku 2.2, čo predstavuje vylepšený MiniMax. Zlepšienie spočíta v tom že pri prehľadávaní stromu s ťahmi sú vynechané vetvy, ktoré neovplyvnia výsledné hodnotenie. Tým je znížená náročnosť prehľadávania a umožnené vyhodnotenie nasledujúceho ťahu v rozumnom čase.

Nasledujúcim použitým algoritmom je Prehĺadávanie iteratívnym prehlbovaním (*Iterative deepening search*). Napriek použitiu Alfa-Beta orezávania, stavový priestor stále exponenciálne narastá, až kým nebola ukončená hra. To mnohokrát vedie ku zaplneniu celej hracej plochy. Riešením tohto problému je ukončenie prehľadávania po dosiahnutí vopred určenej hĺbky. Tu nastáva ďalší problém. Prehľadávanie do určenej hĺbky samo o sebe nemusí vždy fungovať, pretože hĺbka nemusí byť dostatočná, prípadne môže byť zbytočne veľká a jednoduché rozhodnutia by trvali dlhšie, ako je potrebné. Preto sa začína s malou hĺbkou a ak ja jedno prehľadanie vykoná do určenej časovej hranice, hĺbka sa inkrementuje a prehĺadanie sa opakuje.

Posledným použitým algoritmom je Stromové prehľadávanie Monte Carlo (*Monte Carlo tree search*, ďalej MCTS). Algoritmus je založený na náhodnom prehľadávaní stavového priestoru. V jednom priebehu algoritmu je generovaný strom, ktorého uzly reprezentujú stavy hracej plochy a hrany reprezentujú jednotlivé ťahy. Po skončení hry sa inkrementuje počet navštívenia uzlu pre všetky vygenerované uzly. Ak hra skončila výhrou, inkrementuje sa počet výhier pre každý vygenerovaný uzol. Pomocou týchto hodnôt sú vyberané nasledujúce uzly pre rozgenerovanie a nakoniec ohodnotené jednotlivé ťahy. Prehľadávanie je podrobne popísané v bčlánku [1].

2.3 Výsledky porovnania

Jednotlivé hry boli hrané na klasickej hracej doske o veľkosti 19x19 políčok pričom 5 žetónov v rade bolo potrebných pre výhru. Časovo ohraničené algoritmy boli limitované na 15 sekúnd a MCTS bolo obmedzené na vyskúšanie 2000 ťahov. Pravidlá boli rovnaké ako pri voľnej hre, kedy je zvýhodnený prvý hráč. Porovnania jednotlivých algoritmov dopadli nasledovne:

- Heuristika vs. hĺbhovo limitovaný Minimax: V tomto prípade vždy vyhral algoritmus, ktorý ťahal ako prvý. Toto súhlasí s tvrdením, že pri voľnej hre je zvýhodnený prvý hráč.
- Heuristika vs. Iteratívne prehlbovanie: Nezávisle na tom ktorý algoritmus ťahal prvý, vždy vyhral algoritmus Iteratívneho prehlbovania.
- Heuristika vs. MCTS: Keďže heuristika nedokázala poraziť Iteratívne prehlbovanie ani raz, proti Stromovému prehľadávaniu Monte Carlo taktiež vyhrať nedokázala, presne ako bolo očakávané.
- Hĺbkovo limitovaný MiniMax vs. Iteratívne prehlbovanie: Keďže Iteratívne prehlbovanie je vylepšenie nad samotným MiniMax algoritmom, je očakávané že Iteratívne prehlbovanie bude vyhrávať, čo bolo potvrdené experimentami.
- Iteratívne prehlbovanie vs. MCTS: Experimenty ukázali, že Iteratívne prehlbovanie takmer konzistentne porazí MCTS. Problémom bolo, že porazený algoritmus potrebuje dostatočne silnú heuristickú funkciu. Použitá heuristická funkcia mohla stačiť

pre Iteratívne prehlbovanie, avšak pre MCTS to nestačilo. Pozorovaním sa zistilo, že MCTS sa pokúša hrať o pár žetónov ďalej od hlavnej skupiny žetónov. Túto taktiku zvolil zrejme kvôli tomu, aby mal viac možností počas nasledujúcich ťahov. Narozdiel od toho algoritmus Iteratívneho prehlbovania hrá žetóny bližšie ku ostatným.

- Iteratívne prehlbovanie vs. MCTS: Kvôli tomu, že predošlý experiment nedopadol podľa očakávaní, autor článku nechal oba algoritmy hrať proti sebe na hracej ploche o veľkosti 9x9, pričom pre výhru je nutné uložiť vedľa seba štyri žetóny v ortogonálnom alebo diagonálnom smere. V tomto prípade vždy vyhral hráč, ktorý hral ako prvý. Autor pôvodného článku je presvedčený, že v prípade že by bol algoritmus MCTS vylepšený, dokázal by poraziť algoritmus Iteratívneho prehlbovania. Dôležitým poznatkom je, že kvalita ťahov MCTS algoritmu silne závisí na použitej heuristike pre výber ťahu.

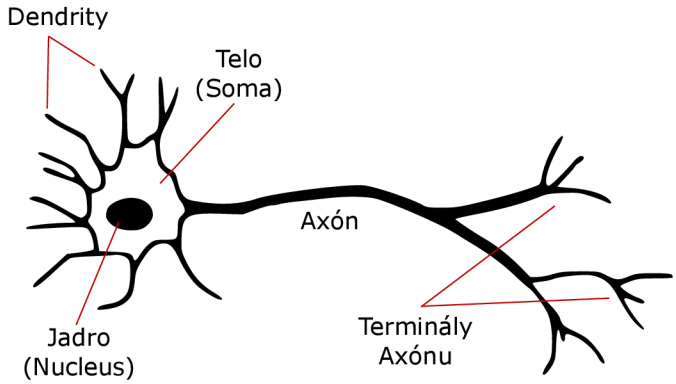
Kapitola 3

Neurón

Nasledujúca kapitola obsahuje popis modelu neurónu, jeho podobnosti s biologickým neurónom a používané aktivačné funkcie. Informácie o neuróne a jeho modele som čerpal z [11].

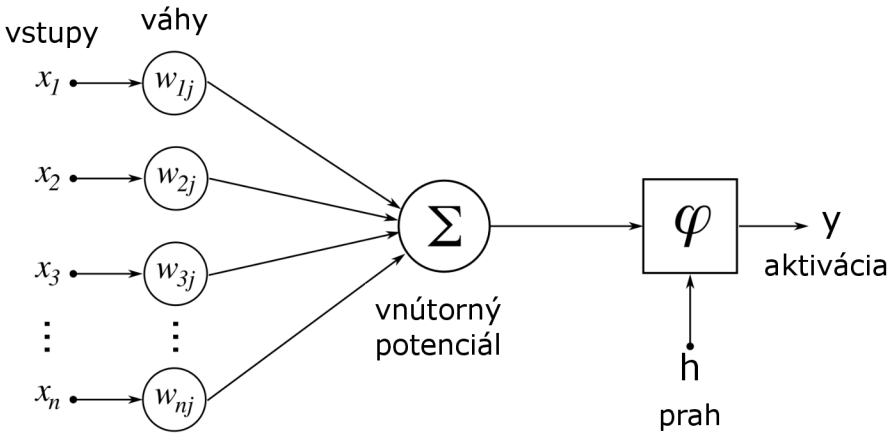
3.1 Biologický neurón

Základným stavebným funkčným prvkom nervovej sústavy je nervová bunka, tzv. *neurón*. Je to samostatná špecializovaná bunka určená k prenosu, spracovaniu a uchovaniu informácii nutných pre realizáciu životných funkcii organizmu. Neurón je prispôsobený pre prenos signálov tak, že okrem vlastného tela, tzv. *somatu*, ma aj vstupné a výstupné prenosové kanály, nazvané *dendrity* a *axón*. Z axónu vybočuje rada tzv. *terminálov*. Ku prenosu informácie slúži unikátne rozhranie medzi neurónmi, tzv. *synapsia*.



Obr. 3.1: Biologický model neurónu. Terminály axónu sa napájajú na dendrity ďalších neurónov, čo umožňuje prenos informácii v mozgu. Prevzaté z https://www.boundless.com/psychology/textbooks/boundless-psychology-textbook/learning-7/biological-basis-of-learning-49/habituation-sensitization-and-potentiation-204-12739/.

Šírenie informácie je možné, pretože telo neurónu a axón sú obalené membránou, ktorá za istých okolností generuje elektrické impulzy. Tieto impulzy sú z axónu prenášané na dendrity iných neurónov tzv. *synaptickými bránami*. Synaptické brány majú istú priepustnosť,



Obr. 3.2: Model formálneho neurónu.

ktorá určuje intenzitu podráždenia ďalších neurónov. Podráždené neuróny po dosiahnutí tzv. *prahu* samé generujú impulz a tým šíria informácie ďalej. Priepustnosť sa po každom priechode signálu mení, čo je predpokladom pamäťovej schopnosti neurónov.

3.2 Model neurónu

Základom matematického modelu neurónovej siete bude *formálny neurón* ilustrovaný na obrázku 3.2. Formálny neurón (ďalej neurón) je získaný zjednodušeným preformulovaním biologického neurónu do matematickej reči. Neurón má všeobecne n reálnych vstupov $x_1, ..., x_n$. Tieto vstupy modelujú jednotlivé dendrity. Vstupy sú ohodnotené svojimi odpovedajúcimi synaptickými váhami $w_1, ..., w_n$, ktoré určujú ich priepustnosť. Synaptické váhy môžu byť aj záporné, čím sa vyjadruje ich inhibičný charakter. Suma vstupov prenásobených prislúchajúcimi váhami predstavuje *vnútorný potenciál* neurónu ξ. Hodnota vnútorného potenciálu po dosiahnutí tzv. *prahovej* hodnoty h indukuje výstup neurónu y, ktorý modeluje elektrický impulz axónu. Nelineárny nárast výstupnej hodnoty pri dosiahnutí prahovej hodnoty je daný tzv. *aktivačnou* (prenosovou) funkciou. Výstup danej aktivačnej funkcie vyjadruje, či neurón rozpoznal kombináciu vstupov, na ktorú sú jeho váhy natrénované.

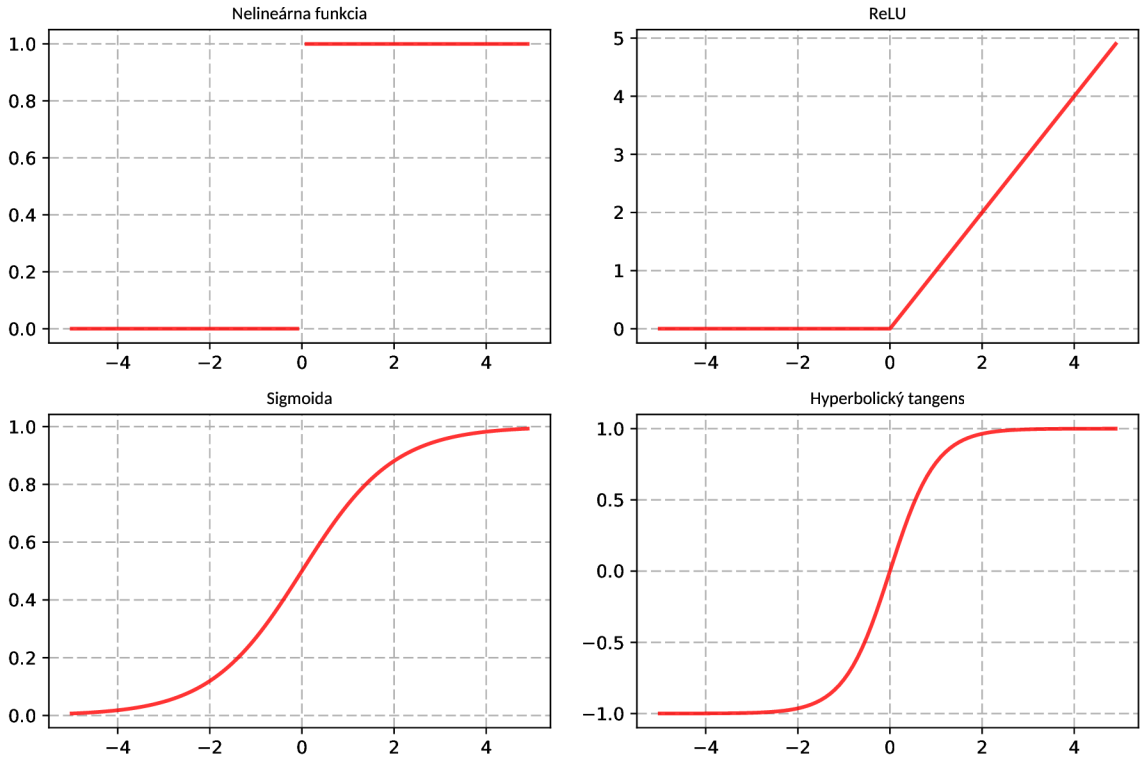
3.3 Aktivačné funkcie

Najjednoduchším typom aktivačnej funkcie je tzv. *ostrá nelinearita*, ktorá má tvar

$$f(\xi) = \begin{cases} 1 & \text{pre } \xi \geq h \\ 0 & \text{pre } \xi < h \end{cases} \tag{3.1}$$

Po formálnej úprave bude mať funkcia nulový prah a vlastný prah neurónu so záporným znamiekom budeme chápať ako váhu, tzv. *bias* $w_0 = -h$ ďalšieho formálneho vstupu $x_0 = 1$ s konštantnou jednotkovou hodnotou. Funkcia neurónu je potom daná vzťahom

$$f(\xi) = \begin{cases} 1 & \text{pre } \xi \geq 0 \\ 0 & \text{pre } \xi < 0 \end{cases}, \text{kde } \xi = \sum_{i=0}^{n} w_i x_i \tag{3.2}$$



Obr. 3.3: Grafy funkcii 3.1, 3.3 a iných používaných aktivačných funkcii.

Medzi používané aktivačné funkcie taktiež patrí tzv. *Rectified Nonlinear Unit* (ReLU)

$$ReLU(\xi) = \begin{cases} \xi & \text{pre } \xi > 0 \\ 0 & \text{inak} \end{cases} \tag{3.3}$$

V neposlednej rade sa medzi používané aktivačné funkcie radí sigmoida, hyperbolický tangens a iné. Spomínané aktivačné funkcie sú ilustrované na obrázku 3.3.

Kapitola 4

Od neurónov k neurónovým sieťam

V nasledujúcej kapitole je popísaný vznik neurónových sietí z neurónov, ich rôzne vlastnosti a jednotlivé topológie podľa typu siete.

4.1 Umelé neurónové siete

Zhlukovaním neurónov do vrstiev a ich prepájaním vznikajú neurónové siete. V nich sú neuróny prepojené tak, že výstup neurónu je vstupom viacerích neurónov. Podobným spôsobom sú terminály axónu biologického neurónu spájané pomocou synaptických väzieb s dendritmy iných neurónov. Počet neurónov a ich vzájomné určuje tzv. *architektúru* neurónovej siete. Z hľadiska využitia rozdelujeme neuróny v sieti na:

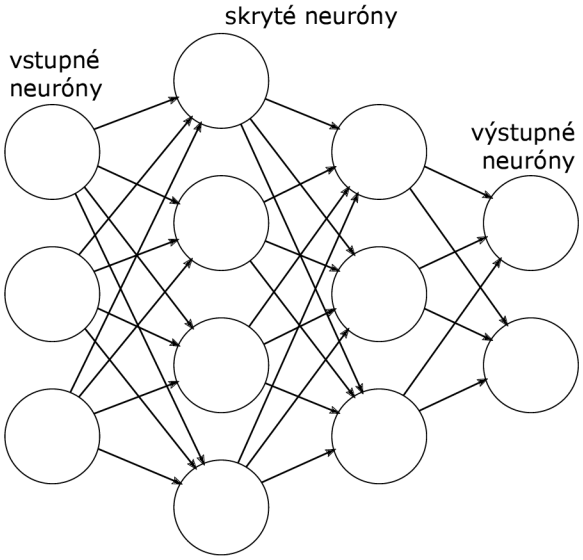
- *vstupné*, ktoré sú vždy minimálne v prvej vrstve a obsahujú vstupné dáta neurónovej siete
- *skryté*, ktoré sú medzi vstupnými a výstupnými, sú trénované a vykonávajú určený výpočet
- *výstupné*, ktoré sú vždy minimálne na konci neurónovej siete a po dokončení výpočtu obsahujú výstupné dáta

Stav všetkých neurónov určuje tzv. *stav* neurónovej siete a synaptické váhy všetkých spojov predstavujú tzv. *konfiguráciu* neurónovej siete. Časom sa neurónová sieť vyvýja, mení sa prepojenie neurónov, stav neurónov a váhy sa adaptujú. Kvôli tomu sa *dynamika* neurónovej siete delí na tri skupiny a uvažujú sa tri režime práce siete. Organizačná dynamika popisuje zmena topológie, aktivívna dynamika vyjadruje zmenu stavu a adaptívna spôsobuje zmenu konfigurácie.

4.2 Delenie architektúr

Ako prvé sa architektúry neurónových sietí delia na:

- *cyklické* (resp. *rekurentné*), v ktorých existuje skupina neurónov zapojených v kruhu tak, že výstup posledného neurónu zo skupiny je vstupom prvého neurónu
- *acyklické* (resp. *dopredné*), v ktorých cyklus neexistuje a všetky cesty vedú jedným smerom



Obr. 4.1: Príklad architektúry 3-4-3-2 a jednotlivých typov neurónov v neurónovej sieti z hľadiska využitia. Čísla v označení architektúry vyjadrujú počet neurónov v jednotlivých vrstvách od vstupnej po výstupnú.

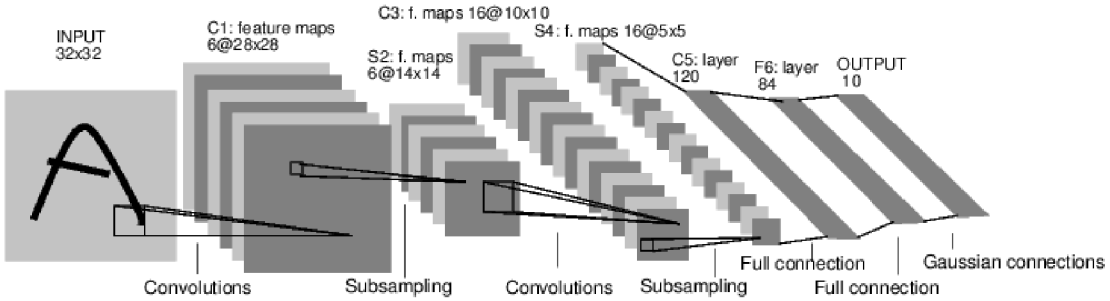
Jednoduchým príkladom cyklu v neurónovej sieti je *spätná väzba* neurónu, ktorého výstup je zároveň jeho vstupom. Topológia, ktorá obsahuje najviac cyklov, sa nazýva *úplná topológia*.

Pri acyklických sieťach je možné neuróny rozdeliť do tzv. *vrstiev* usporiadaných tak, že spoje medzi neurónmi vedú iba z nižších vrstiev do vyšších. Vo všeobecnom prípade môžu preskočiť jednu alebo viac vrstiev. Vrstvy sa číslujú od nuly, teda $n + 1$ vrstiev tvorí n-vrstvovú sieť. Je to kvôli tomu, že vstupná vrstva sa do počtu vrstiev siete nepočíta.V topológii viacvrstvovej siete sú neuróny jednej vrstvy spojené so všetkými neurónmi nasledujúcej vrstvy. Chýbajúce spoje sa dajú chápať ako spoje s váhou rovnou nule. Preto architektúru takejto siete vieme popísať iba počtami neurónov, spravidla oddelenými pomlčkou, v poradí od vstupnej po výstupnú vrstvu. Pomocou tohto pravidla je označená aj neurónová sieť na obrázku 4.1.

4.3 Použitie neurónových sietí

Neurónové siete je možné použiť na riešenie viacerích typov úloh, medzi ktoré patrí napr. klasifikácia, aproximácia, segmentácia a iné. Medzi známe využitia neurónových sietí patria aj tieto:

- Neurónová sieť pre klasifikáciu [5] 1000 tried, natrénovaná na dátovej sade LSVRC-2010 ImageNet, na testovacej sade dosiahla presnosť 18.9%, čo bolo lepšie ako doposiaľ dosiahnuté výsledky.
- LeNet-5 [6] je neurónová sieť určená na rozpoznávanie znakov v texte, natrénovaná na MNIST trénovacej sade, s presnosťou približne 99%.
- Neurónové siete na prenos umeleckého štýlu [2], ktoré z obrazu maliara extrahujú štýl a aplikujú ho a iný obrázok môžu do istej miery nahradiť moderných umelcov.



Obr. 4.2: Architektúra neurónovej siete LeNet-5. Neurónová sieť slúži na klasifikáciu písmen a čísel. Prevzaté z http://eblearn.sourceforge.net/beginner_tutorial2_train.html.

- Program AlphaGo [9] porazil európskeho šampióna v hre Go so skóre 5:0. Toto bolo prvý krát, keby počítačový program prorazil profesionálneho hráča v tejto hre.
- Predpovedanie budúcnosti na základe histórie, napríklad vývoj cien na marketingovom trhu. Perfektná predikcia nie je reálna, avšak dosiahnuteľná predikcia je mnohokrát postačujúca.

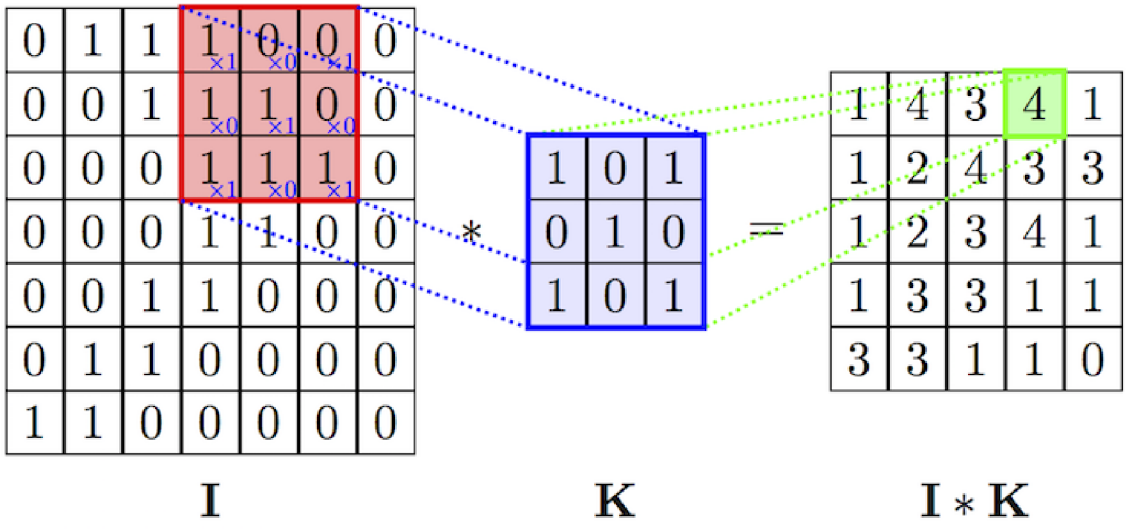
Kapitola 5

Konvolučné neurónové siete

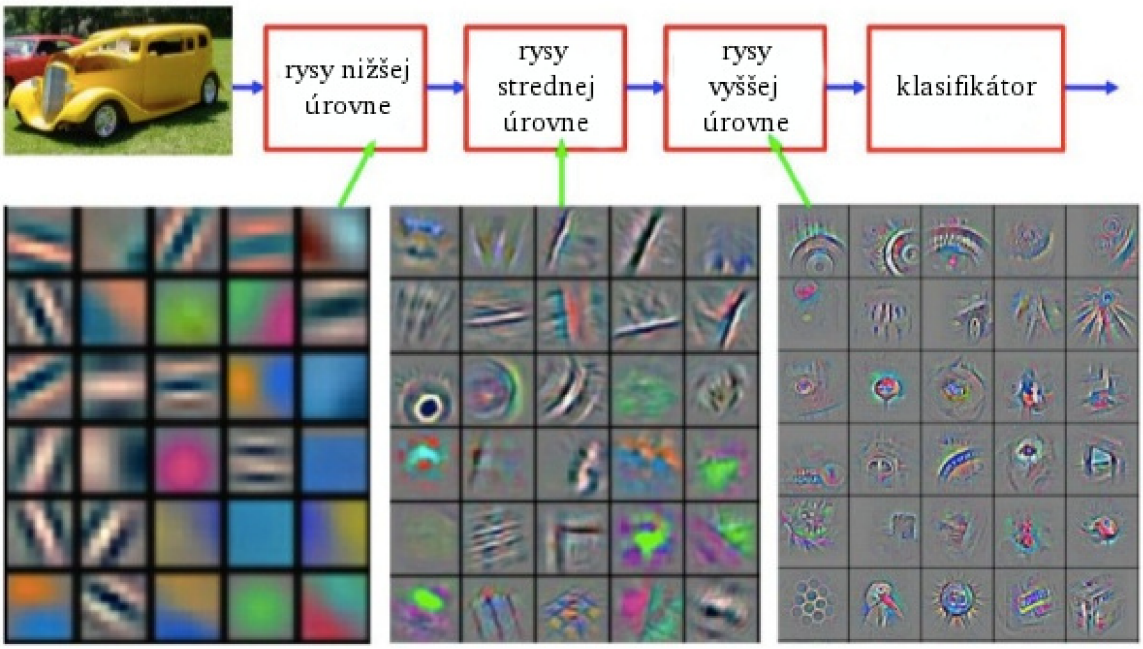
Rovnako ako iné neurónové siete, konvolučné neurónové siete pozostávajú z neurónov, ktoré sú zoskupené do vrstiev. Tiež majú chybovú funkciu za poslednou vrstvou. Kvôli tomu, že vstupom sú väčšinou obrázky, majú architektúry iné vlastnosti, ktoré sú popísané ďalej, v tejto kapitole. Takto je dopredný prechod efektívnejší a počet parametrov siete je redukovaný.

5.1 Vlastnosti konvolučných neurónových sietí

Bežná neurónová sieť vstupný vektor trasfrormuje pomocou skrytých vrstiev. Každá vrstva pozostáva z neurónov, z ktorých každý je prepojený so všetkými neurónmi z predošlej vrstvy. Problém je, že neurónové siete, ktorých architektúra pozostáva z viacerích plne prepojených vrstiev, majú veľký počet vrstiev aj pri relatívne malých obrázkoch na vstupe.



Obr. 5.1: Ilustrácia výpočtu konvolučnej vrstvy pri vstupe I a konvolučnom jadre K. Prevzaté z https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html.



Obr. 5.2: Vizualizácia jednotlivých rysov, pre ktorých identifikáciu sú natrénované jednotlivé konvolučné jadrá konvolučných vrstiev. Prvé konvolučné vrstvy detekujú iba veľmi jednoduché rysy na nízkej úrovni ako hrany, prípadne jednoduché krivky. Ďalšie konvolučné vrstvy reagujú na komplikovanejšie vzory ako rôzne kruhy s krivkami. Posledné konvolučné vrstvy detekujú komplikovanejšie obrazce. Prevzaté z https://m2dsupsdlclass.github.io/lectures-labs/slides/03_conv_nets/index.html.

Pri obrázku o veľkosti 200x200x3 by neuróny mali $200 * 200 * 3 = 120\,000$ váh. Okrem toho, že by výpočet trval niekoľko krát dlhšie ako je nutné, takýto počet váh a nadmerné prepojenie jednotlivých neurónov by pravdepodobne viedli k pretrénovaniu.

Namiesto toho má každá vrstva niekoľko konvolučných jadier s vopred určenou veľkosťou. Jednotlivé hodnoty konvolučných jadier sú trénované pri spätnom priechode podľa gradientu vypočítaného. *Výplň* (padding) slúži na ohraničenie vstupného tenzoru, čo umožnuje limitovať zmenšenie výstupu konvolúcie. Pre viacnásobné zmenšenie výstupu konvolúcie je možné zmeniť hodnotu *kroku*. Táto hodnota ovplyvňuje posun konvolučného jadra po každej konvolúcii. V prípade že chceme vypočítať konvolúciu pre všetky možné pozície, hodnota kroku je rovná 1.

5.2 Konvolučné neurónové siete v umelej inteligencii

Problémom, ktorý je podobný rozoberanému problému v tejto práci, sa zaoberala spoločnosť DeepMind. Ich cieľom bolo vytvoriť umelú inteligenciu hrajúcu hru Go. Táto hra sa hrá na rovnako veľkej hracej ploche ako Gomoku, avšak pravidlá sú komplexnejšie a zakladajú sa na ohraničovaní plochy pomocou žetónov. Nakoniec sa im podarilo vytvoriť umelú inteligenciu zvanú AlphaGo [9], ktorá porazila európskeho šampióna so skóre 5:0. Bolo to prvý krát, kedy umelá inteligencia porazila profesionálneho hráča Go. Ich riešenie si zakladá na použití dvoch neurónových sietí a ich zakomponovaní do Stromového prehľadávania Monte Carlo.

Prvým krokom pri tvorbe umelej inteligencie bolo trénovanie strategickej siete na hracích plochách s príslušným ťahom pomocou stochastického gradientného vzostupu. Táto neurónova sieť dostane na vstup aktuálnu konfiguráciu hracej plochy a jej výstupom je distribúcia pravdepodobnosti pre všetky pozície na hracej ploche. Architektúra strategickej siete pozostávala z 13 konvolučných vrstiev a na trénovanie bolo použitých 30 miliónov pozícii. Týmto spôsobom dosiahli presnosť 57% pri použití všetkých vstupných rysov. Taktiež bola natrénovaná rýchla strategická sieť, ktorá je menšia a má presnosť iba 24.2%. Výhodou však je, že výber ťahu jej trvá $2\,\mu s$, narozdiel od pôvodnej strategickej siete, ktorej výber ťahu trvá 3 ms. Druhým krokom bolo trénovanie strategickej siete pomocou spätnoväzobného učenia (*reinforcement learning*). Hrajú sa hry medzi aktuálnou strategickou sieťou a náhodnou predošlou iteráciou, aby sa predošli pretrénovaniu. Je použitá funkcia odmeňovania, ktorá je rovna nule pre všetky neterminálne ťahy. Odmena je určená výsledkom hry: +1 pri výhre a -1 pri prehre. Váhy sú aktualizované pomocou stochastického gradientného vzostupu v smere, ktorý maximalizuje očakávaný výsledok. Takto trénovaná neurónová sieť vyhrala 80% hier proti neurónovej sieti trénovanej iba na hracích plochách. Neurónová sieť bola taktiež testovaná pri hraní s programom Pachi, ktorý je založený na MCTS a vykoná 100 000 simulácii za ťah. Bez použitia akéhokoľvek prehľadávania dokázala strategická sieť trénovaná pomocou spätnoväzebného učenia poraziť program Pachi pri 85% zo všetkých prípadov.

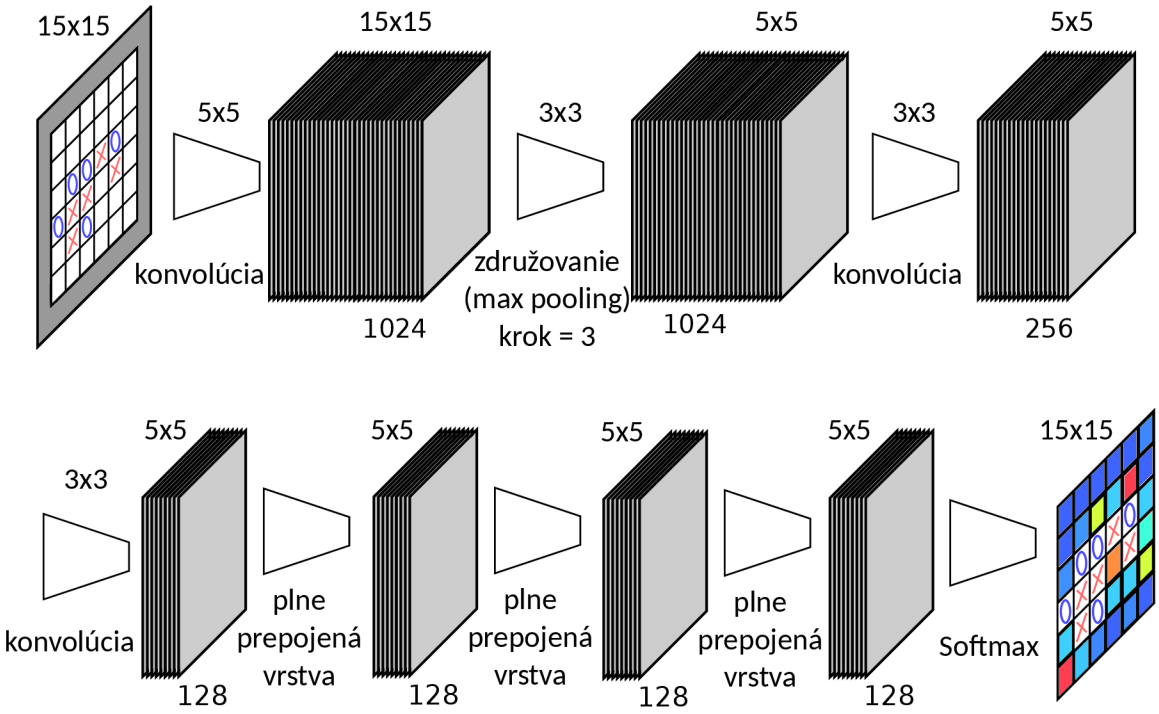
Posledným krokom bolo spätnoväzebné učenie ohodnocovacej siete. Táto neurónová sietť dostáva na vstup konfiguráciu hracej plochy a jej výstupom je pravdepodobnosť výhry pri danej konfigurácii. Pokus o predpovedanie výsledku hry z dát, ktoré pozostávajú z odohraných hier, viedol k pretrénovaniu. Neurónová sieť sa pokúšala zapamätať si výsledky hier. Týmto spôsobom bola dosiahnutná stredná kvadratická chyba (ďalej MSE) 0.37 na testovacej sade a 0.19 na trénovacej sade. Preto bola vytvorená nová dátová sada pozostávajúca z 30 miliónov rôznych pozícii, pričom každá pochádzala z inej hry. Tieto hry hrala medzi sebou strategická sieť trénovaná spätnoväzobným učením. Trénovanie na novej dátovej sade viedlo k MSE o hodnote 0.226 a 0.234 a trénovacej a testovacej sade. Jediné vyhodnotenie ohodnocovacej siete dosiahlo presnosť strategickej siete trénovanej spätnoväzobným učením za použitia 15 000-násobne menej výpočtov.

AlphaGo kombinuje neurónové siete s MCTS algoritmom, ktorý vyberá akciu pomocou prehľadávania stavového priestoru. Stavový priestor je reprezentovaný stromom, pričom každá hrana reprezentuje stav hracej plochy a hraný ťah. Jednotlivé hrany si ukladajú hodnotu ťahu $Q(s,a)$, počet navštívení $N(s,a)$ a pôvodnú pravdepodobnosť $P(s,a)$. Pri každom kroku simulácie t sa vybraný ťah a_t pri stave hracej plochy s_t

$$a_t = argmax_a(Q(s_t,a) + \frac{P(s,a)}{1+N(s,a)} \tag{5.1}$$

pričom cieľom je maximalizovať hodnotu ťahu a bonus. Bonus je ovplyvnený pôvodnou pravdepodobnosťou ale postupne sa zmenšuje s narastajúcim počtom navštívení, čo vedie ku vyhodnocovaniu iných hrán. Stav hracej dosky listového uzlu je vyhodnotený strategickou sieťou. Listový uzol je vyhodnotený ohodnocovacou sieťou a pomocou výsledku hry odohranej pomocou rýchlej strategickej siete. Výsledky sú kombinované pomocou parametru λ, z čoho vychádza hodnota listového uzlu $V(s_L)$

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L \tag{5.2}$$



Obr. 5.3: Ilustrácia konvolučnej neurónovej siete z článku [8]

Po skončení simulácie sa aktualizujú hodnoty ťahu a počty navštívení všetkým navštíveným hranám. Každá hrana akumuluje počet navštívení a priemerný výsledok simulácii, ktoré prechádzali cez danú hranu

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i) \tag{5.3}$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i) \tag{5.4}$$

kde s_L^i je listový uzol $i$tej simulácie a $1(s,a,i)$ indukuje, či hrana (s,a) bola navštívená počas $i$tej simulácie. Po skončení prehľadávania algoritmus vyberá najviac navštívenú hranu z koreňového uzlu.

V článku [10] sa autor pokúšal vytvoriť dve neurónové siete. Prvá z nich bola konvolučná neurónová sieť. Jej architektúra pozostáva z konvolučných vrstiev a plne prepojených vrstiev. Podrobný popis architektúry je na obrázku 5.3. Ako druhá sieť bola použitá rekurentná neurónová sieť. Pri tejto sieti sa autor inšpiroval článkom [8]. Táto neurónová sieť sa mala zamerať na škálovateľnosť herných plôch. Pri experimentoch obe neurónové siete hrali proti naivnej umelej inteligencii, ktorá využívala prehľadávanie stavového priestoru. Pri trénovaní konvolučnej neurónovej siete sa výstup chybovej vrstvy blíži ku 2. Presnosť pri validácii na testovacej dátovej sade bola približne 40%. Neurónová sieť nebola schopná poraziť prehľadávací algoritmus ani jeden krát z tisíc hier. Pozorovanie hier ukázalo, že napriek tomu, že neurónová sieť občas zahrala dobré ťahy, nebola schopná voliť správne ťahy konzistentne. Pri trénovaní rekurentnej neurónovej siete sa výstup chybovej vrstvy blíži ku 1. Táto neurónová sieť si viedla lepšie ako konvolučná neurónová sieť. Presnosť pri validácii dosahovala hodnotu približne 50%, avšak táto neurónová sieť mala taktiež problém

s konzistentnou voľbou správnych ťahov. Proti algoritmu, ktorý využíva stromové prehľadávanie, prehrala v pomere 3:997. Pri priebehu jednej z hier neurónová sieť nezablokovala štyri žetóny protihráča, čo viedlo k prehre. Namiesto toho zahrala nezmyselý ťah.

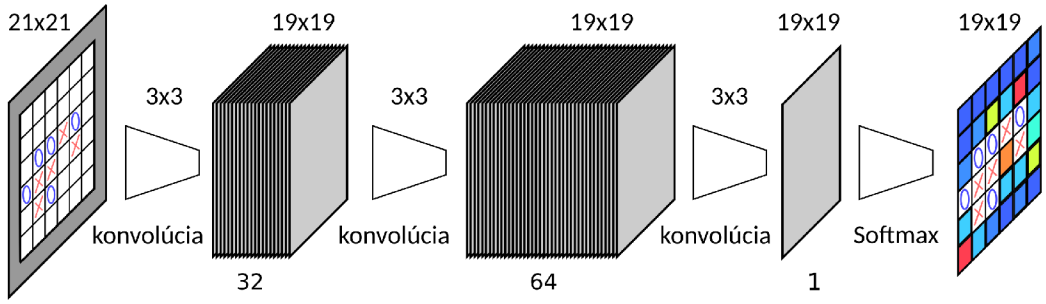
Kapitola 6

Navrhnuté riešenie

V tejto kapitole sú podrobne popísané použité neurónové siete a prehľadávacie algoritmy, ktoré umelá inteligencia používa, pre určenie nasledujúceho ťahu. Konvenčné umelé inteligencie si zakladajú na tom, že prehľadávajú stavový strom. Pri hrách s veľkým stavovým priestorom sú klasické prehľadávacie metódy problémové, pretože je nutné pracovať s veľkým množstvom hracích plôch, čo je v rozumnom čase nedosiahnuteľné. Do úvahy prichádza obmedzenie šírky a hĺbky prehľadávania. Šírka prehľadávania predstavuje počet prehľadaných nových hracích plôch z jednej hracej plochy a hĺbka predstavuje počet vygenerovaných hracích plôch, pričom nasledujúca je vždy vygenerovaná z predošlej. Šírka a hĺbka prehľadávania sú obmedzené strategickou a ohodnocovacou sieťou, ktoré sú popísané v nasledujúcej kapitole. Zdrojové kódy sú umiestnené v repozitári na GitHube[1].

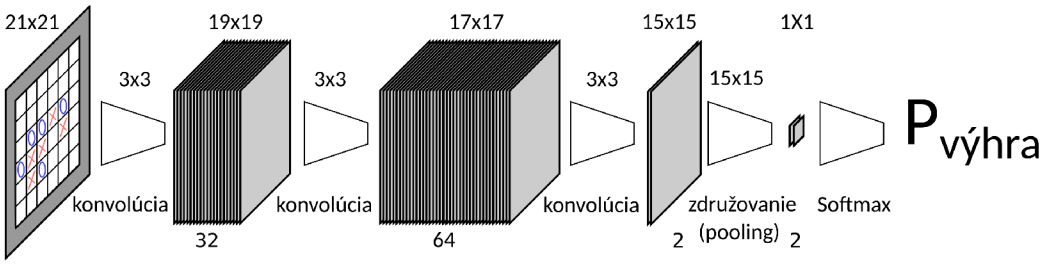
6.1 Architektúry neurónových sietí

Navrhnutá umelá inteligencia využíva dve neurónové siete. Prvá z nich, zvaná *strategická sieť*, dostane na vstup konfiguráciu hracej plochy a jej výstupom je pravdepodobnostné rozloženie pre všetky pozície na ploche. To vytvára tzv. mapu pravdepodobností, ilustrovanú na obrázku 6.1. Neurónová sieť s rovnakou funkciou bola použitá v programe AlphaGo [9].



Obr. 6.1: Architektúra strategickej siete s tromi konvolučnými vrstvami. Vstupná hracia plocha je ohraničená výplňou pre vyznačenie hranice plochy a zachovanie rozmerov. Rovnako je výplň použitá na ohraničenie plochy aj v každej konvolučnej vrstve. Výstupom je pravdepodobnosť ťahu pre všetky políčka ilustrovaná teplotnou mapou. Pôvodné žetóny sú na mape ponechané iba kvôli ilustrácii hodnôt okolitých políčok.

[1] https://github.com/Hacky-MB/destruCtioNN



Obr. 6.2: Architektúra ohodnocovacej siete s tromi konvolučnými vrstvami. Vstupná hracia plocha je ohraničená výplňou pre označenie okraja plochy. Výstupom sú dve hodnoty určujúce pravdepodobnosť výhry.

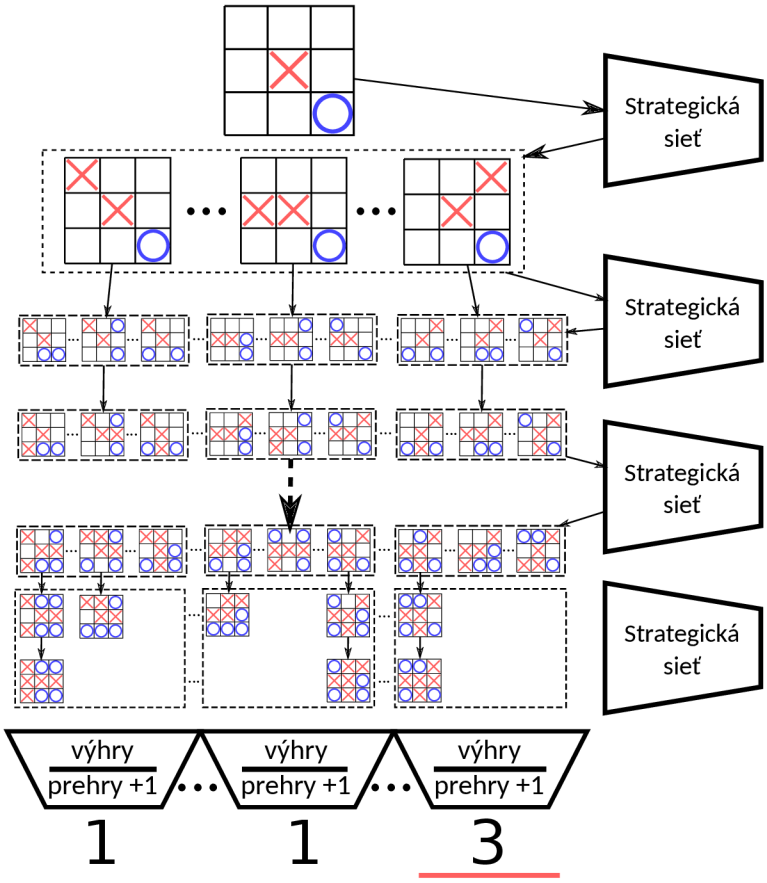
Každá z konvolučných vrstiev strategickej siete, okrem poslednej, je nasledovaná tzv. *ReLU* vrstvou, ktorá reprezentuje aktivačnú funkciu. Po nej nasledovala vrstva pre normalizácu dávky (*batch normalization*) [3]. Posledná konvolučná vrstva je vstupom *Softmax* vrstvy [7], ktorá vypočíta jednotlivé výstupné pravdepodobnosti. Strategické siete som navrhol tak, aby okolie, o ktorom mala informácia každá pozícia, boli minimálne dve políčka z každej strany. Toto je nutné, pretože kľúčovou úlohou je vyhľadávanie pätice kameňov v rade. Ideálne je, aby okolie bolo ešte väčšie, pretože aj políčka v okolí plochy so stranou o veľkosti päť políčok môžu rozhodovať o kvalite danej konfigurácie.

Strategické siete, ktoré sa líšia počtom konvolučných vrstiev, som navrhol tak, aby jedna iterácia trvala približne 30 až 50 milisekúnd. Ich architektúry sú podrobne popísané v tabuľke 6.1. Čas bol zmeraný pri počítaní na grafickej karte a s dávkou, ktorá obsahovala 512 hracích plôch. Približne rovnaká dĺžka iterácie bola dosiahnutá zväčšovaním a zmenšovaním počtu výstupov jednotlivých konvolučných vrstiev.

Ohodnocovacia sieť, ako druhá z dvojice, je podrobne znázornená na obrázku 6.2. Táto neurónová sieť sa taktiež inšpiruje jednou z neurónových sietí použitých v AlphaGo [9].

Typ siete	Okolie	Konvolučné vrstvy					
L2	7	5 256	3 1				
L3	7	3 64	3 128	3 1			
L4	13	3 64	5 64	5 32	3 1		
L5	15	3 32	5 64	5 64	3 32	3 1	
L6	13	3 32	3 64	3 64	3 32	3 32	3 1

Tabuľka 6.1: Popis architektúr jednotlivých strategických sietí. Druhý stĺpec reprezentuje rozmer plochy, ktorá je znázorňuje koľko susediacich políčok ovplyvňuje výslednú predikciu. Horný riadok pravej časti tabuľky obsahuje veľkosť konvolučného jadra a spodný počet výstupov vrstvy.



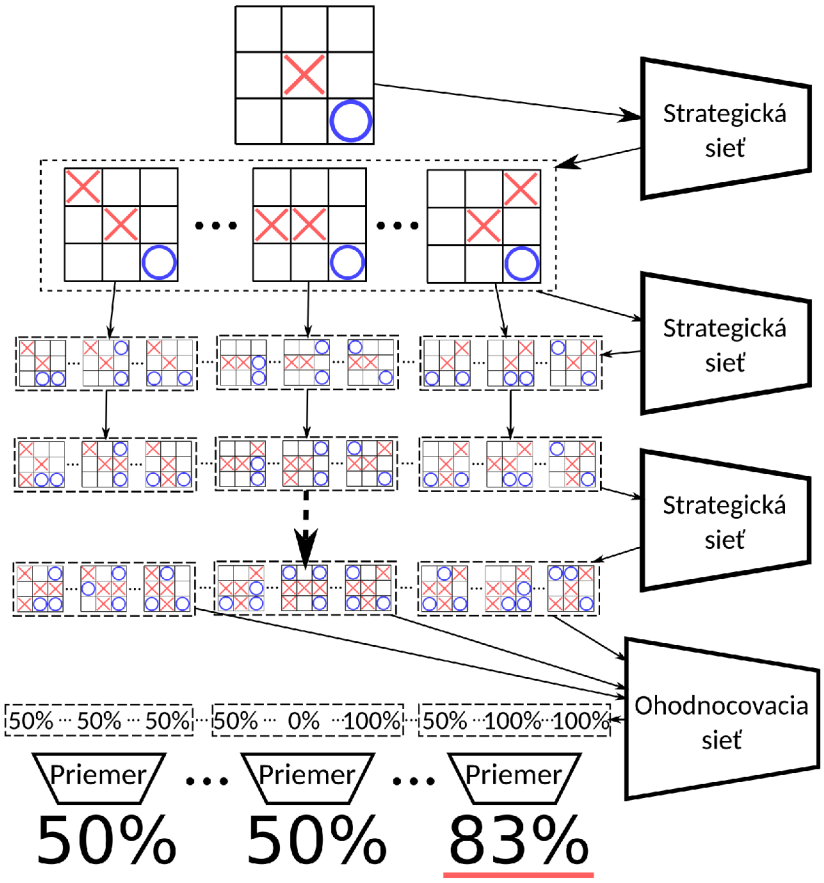
Obr. 6.3: Prehľadávací algoritmus ktorý využíva strategickú sieť pre určenie nasledujúceho ťahu.

Vstupom ohocnocovacej siete je hracia plocha, avšak od strategickej siete sa líši tým, že jej výstupom je pravdepodobnosť výhry pri danej konfigurácii. Hlavný rozdiel medzi týmito sieťami je, že pri strategickej sieti je nutné zachovať veľkosť výstupu, aby nebol menší ako samostatná hracia plocha. To je dosiahnuté používaním výplne v konvolučných vrstvách. Konvolučné a združovacie (*pooling*) vrstvy v architektúre ohodnocovacej sieti vstupné dáta postupne zmenšujú až na konečný rozmer 1x1, čo predstavuje výslednú pravdepodobnosť.

Vstupom oboch sietí je pole s dvomi kanálmi, pričom prvý kanál reprezentuje vlastnú hraciu plochu a druhý kanál hraciu plochu protihráča. Zabrané miesto v poli má hodnotu 1 a voľné políčka sú označené hodnotou 0. Jednotlivé kanály reprezentujú pole o veľkosti 19x19 ohraničené výplňou s hodnotou -1. Šírka výplne je vypočítaná pomocou vzorca $\lfloor k/2 \rfloor$, kde k je veľkosť konvolučného jadra prvej konvolučnej vrstvy. Neurónová sieť potrebuje mať informáciu o tom, či sú dané pozície pri okraji, alebo nie. Preto okraj musí byť odlíšený od ostatných hodnôt. Zároveň je však nutné zachovať rozmer plochy 19x19, čo daný vzorec spĺňuje.

6.2 Prehľadávacie algoritmy

Pri vyhľadávaní sú použité dva algoritmy, ktoré sa inšpirujú algoritmom, použitým v programe AlphaGo. Oba začínajú tým, že vezmú prvotnú konfiguráciu hracej plochy a pomo-



Obr. 6.4: Prehľadávací algoritmus ktorý využíva strategickú a ohodnocovaciu sieť pre určenie nasledujúceho ťahu.

cou strategickej siete vygenerujú prvotný ťah. V prípade, že si to predstavíme ako strom, konfigurácie hracej plochy predstavujú uzly a jednotlivé ťahy, z ktorých vznikajú ďalšie konfigurácie, predstavujú hrany.

Prvá metóda, ilustrovaná na obrázku 6.3, využíva iba strategickú sieť. Z vybraného počtu prvotných ťahov prevedie simuláciu až do ukončenia všetkých simulovaných hier. Z každého vybraného prvotného ťahu vytvorí novú konfiguráciu nastavením hodnoty políčka podľa súradníc ťahu. Vytvorenú konfiguráciu nakopíruje toľko krát, aký je parameter šírky vyhľadávania. Konfigurácia je ďalej vstupom strategickej siete, ktorá ďalej generuje ťahy a vybraný ťah aplikuje na príslušnú hraciu plochu. Každý ťah je vybraný s pravdepodobnosťou určenou strategickou sieťou. Simulácia končí, keď je na všetkých konfiguráciách splnená podmienka ukončenia hry. Výsledné hodnotenie prvotného ťahu je pomer výhier a prehier. Druhá metóda, ilustrovaná na obrázku 6.4 sa od predošlej líši tým, že simulácia prebieha do určenej hĺbky a aproxmácia hodnotenia je určená pomocou ohodnocovacej siete. Narozdiel od AlphaGo, tieto metódy nepoužívajú zľavový parameter, ktorý zvyšuje kvalitu hier ukončených skôr.

Kapitola 7

Experimenty

Pre vyhodotenie kvality navrhnutej umelej inteligencie som vytvoril databázu unikátnych hier zaznamenávaním hraných hier medzi programami zúčastnenými v súťaži Gomocup. Vytvorená dátabáza bola na použitá na trénovanie neurónových sietí navrhnutých v kapitole 6.1. Vyhodnotené experimenty sú popísané v nasledujúcich podkapitolách.

7.1 Použitá knižnica

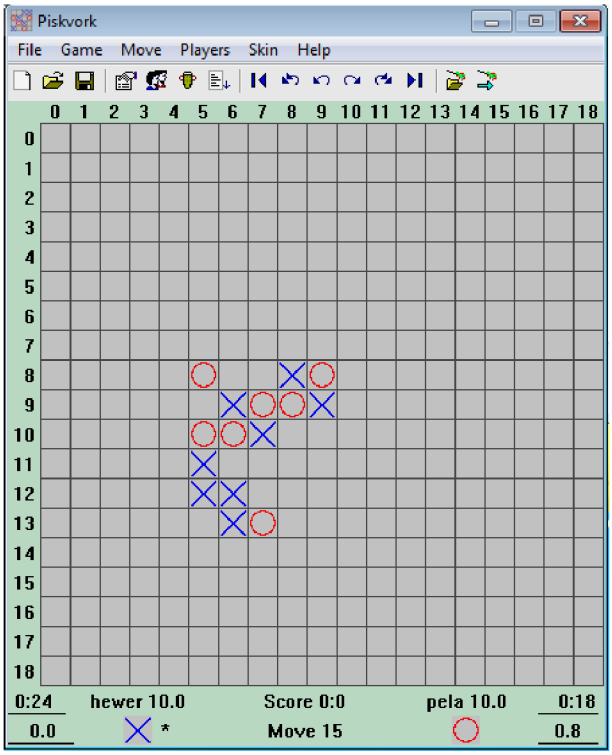
Pre prácu s neurónovými sieťami som zvolil knižnicu Caffe [4]. Na vývoji pracuje Berkeley Vision and Learning Center (BVLC) v spolupráci s neustále prispievajúcimi vývojármi na GitHube. Aj vďaka tomu patrí Caffe ku špičke. Knižnica disponuje nasledujúcimi vlastnosťami:

- Rýchlosť. Knižnica je napísaná v C++ a používa platformu CUDA pre akceleráciu za pomoci grafických procesorov. Na grafickom procesore Titan dokáže spracovať obrázok za približne 2.5 milisekundy.
- Jednoduchosť použitia. Caffe je možné preložiť aj s podporou pre Python alebo MATLAB, čo zjednodušuje prácu s knižnicou. Architektúru neurónovej siete je možné navrhnúť v kóde programu alebo načítať z externého súboru formátu prototxt (plaintext protocol buffer schema). Obsahom knižnice je taktiež skript, ktorý vizualizuje architektúru z .prototxt súboru.
- Podpora iných nástrojov a jazykov. Knižnicu je možné ju preložiť tak, aby podporovala nástoj MATLAB a programovací jazyk Python, čo zjednodušuje prácu s neurónovými sieťami.
- Podpora viacerích vstupných formátov. Vstupné dáta je možné kopírovať priamo do vstupných vrstiev alebo načítavať z HDF5 súborov, prípade LMDB alebo LevelDB databáz. Knižnica taktiež poskytuje nástroje pre konverziu obrázkov do LMDB alebo LevelDB databáz.

7.2 Herný klient

Pri experimentoch som na porovnávanie jednotlivých programov používal herný klient Piskvork[1], ktorý vytvoril Petr Lastovička. Herný klient podporuje hru po sieti a aj na jednom

[1] http://gomocup.org/download-gomocup-manager/

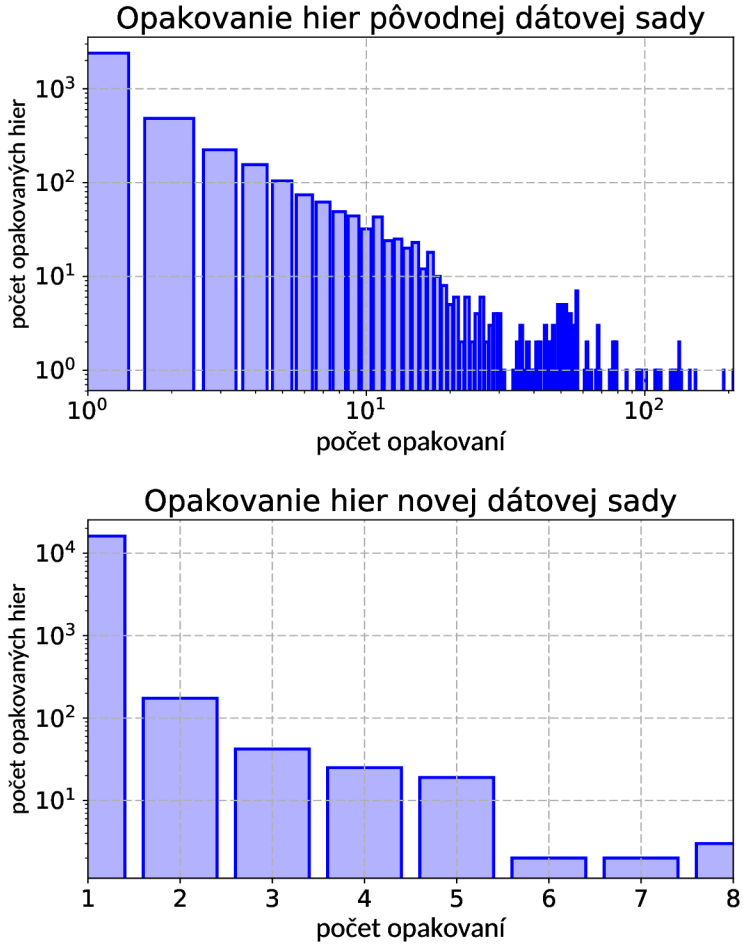


Obr. 7.1: Vzhľad herného klientu Piskvork, ktorý je použitý pri experimentoch.

počítači. Hrať môžu proti sebe dvaja ľudia, človek proti programu a aj dva programy proti sebe. Pri hre viacerích programov proti sebe je možné spustiť turnaj, v ktorom hrajú jednotlivé programy proti sebe určený počet hier. Herný klient taktiež umožnuje limitovať čas určený na ťah programu a celkový čas programu pre celý priebeh hry. Všetky tieto hry je možné zaznamenávať, ukladať a neskôr spracúvať. Pri hre človeka s programom je možné jednotlivé ťahy dajú aj vracať dozadu a prípadne naspäť dopredu, čo pridáva možnosť si každý ťah premyslieť a lepšie odskúšať schopnosti daného programu.

7.3 Dátová sada

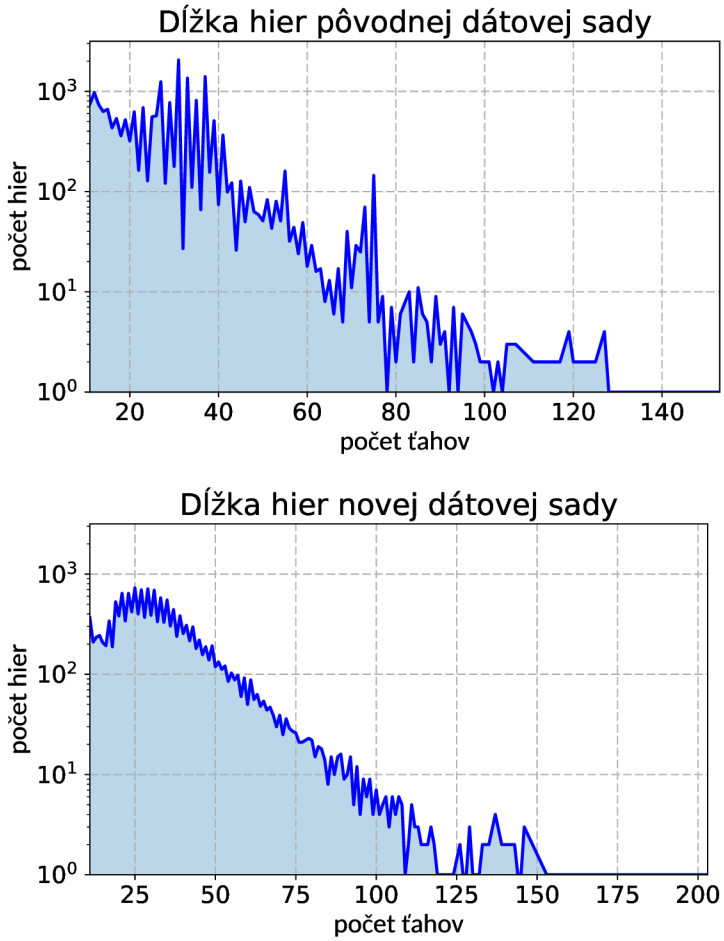
Dáta pre trénovanie neurónových sietí boli získané ukladaním hier hraných medzi stiahnutými programami. Vybrané programy sú umiestnené v rebríčku na prvom až dvadsiatom mieste. Pri generovaní dátovej sady boli súperiace programy rôzne kombinované kvôli zvýšeniu variability hier, avšak to nestačilo. Pôvodne bolo vygenerovaných 19 753 hier a iba 3 953 bolo unikátnych. Problém bol, že všetky hry začínali na prázdnej hracej doske, a preto vzniklo málo unikátnych hier. Situáciu vyriešila možnosť začať hru z uloženej konfigurácie a náhodne ju otáčať. To umožnilo zaznamenať 16 831 hier, z ktorých je 16 379 unikátnych. Porovnanie dĺžky a opakovaní jednotlivých hier pôvodnej a novej dátovej sady je popísané na obrázku 7.2 a 7.3.



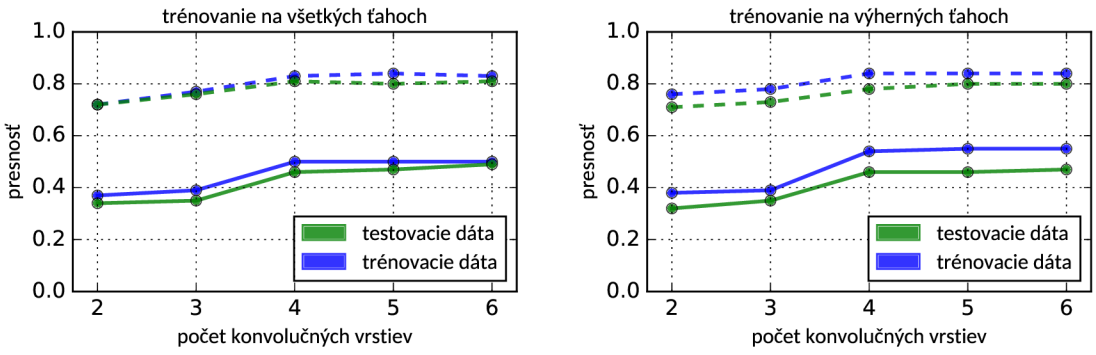
Obr. 7.2: Histogram opakovaní hier v pôvodnej a novej vygenerovanej dátovej sade. Pri generovaní novej dátovej sady boli začiatky hier náhodné, čo zabezpečilo minimalizáciu opakovania jednotlivých hier.

program : sieť	L2	L3	L4	L5	L6
Imro	96:4	94:6	50:50	63:37	50:50
	60:20	8515	58:42	58:42	50:50
Mentat	93:7	87:12	55:45	47:53	30:70
	94:6	88:11	50:50	49:51	45:55
Gmotor	98:2	95:5	93:7	88:12	90:10
	98:2	96:4	91:9	91:9	93:7

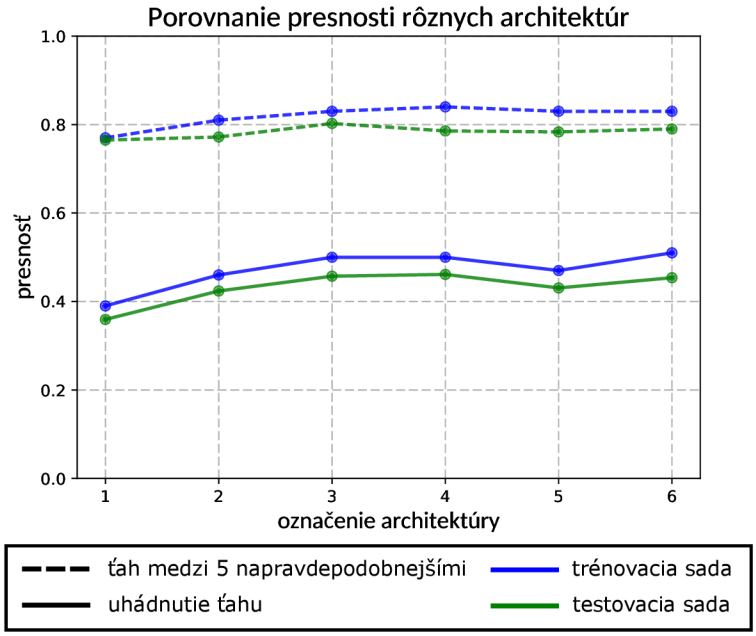
Tabuľka 7.1: Výsledky hier stiahnutých programov proti strategickej sieti. Na hornom riadku z dvojice je výsledok so sieťou natrénovanou na všetkých ťahoch a na spodnom výsledok siete trénovenanej iba na ťahoch, ktoré viedli k výhre. Imro je v tabuľke súťaže Gomocup na 32. , Mentat na 27. a Gmotor na 12. mieste.



Obr. 7.3: Histogram dĺžok hier starej a novej vygenerovanej dátovej sade. Pri generovaní novej dátovej sady boli začiatky hier náhodné, čo zabezpečilo minimalizáciu opakovania jednotlivých hier.



Obr. 7.4: Úspešnosť jednotlivých strategických sietí na testovacej a trénovacej dátovej sade. Plná čiara reprezentuje uhádnutie ťahu a prerušovaná, že hľadaný ťah bol medzi piatimi najpravdepodobnejšími. Kvôli nedostatočnej veľkosti trénovacej dátovej sady sú niektoré neurónové siete mierne pretrénované.



Obr. 7.5: Presnosť jednotlivých architektúr strategickej siete s tromi konvolučnými vrstvami na trénovacej a testovacej sade. Architektúry použitých neurónových sietí sú podrobne popísané v tabuľke 7.2.

označenie architektúry	okolie	rozmery konvolučných jadier		
1	7x7	3x3	3x3	3x3
2	9x9	5x5	3x3	5x5
3	11x11	5x5	5x5	3x3
4	11x11	3x3	5x5	5x5
5	11x11	3x3	3x3	7x7
6	9x9	3x3	7x7	3x3

Tabuľka 7.2: Popis architektúr strategických sietí s tromi konvolučnými vrstvami.

7.4 Strategická sieť

Prvým experimentom je trénovanie strategickej siete na všetkých ťahoch a na ťahoch hier, ktoré skončili výhrou. Trénovanie prebiehalo na viaceríchch architektúrach neurónových sietí ktoré majú od dvoch po šesť konvolučných vrstiev. Podrobnejší popis architektúr sa nachádza v tabuľke 6.1. Úspešnosť trénovania je podrobnejšie popísaná na obrázku 7.4 a ukazuje, že ideálne je použitie architektúry s minimálne štyrmi konvolučnými vrstvami. Experimenty zaznamenané v tabuľke 7.1 ukazujú, že strategická sieť je schopná konkurovať programom v strede tabuľky súťaže Gomocup.

7.5 Strategická sieť s výsledkami

Nasledujúcim experimentom je trénovanie strategickej siete s použitím výsledkov. Ťahy pre trénovanie sú vyberané tak, aby všetky ťahy vybraného vzorku pri trénovaní pochádzali z

hier, ktoré skončili výhrou alebo prehrou. Pri trénovaní bol spustený dopredný priechod, pri ktorom sa vypočítajú gradienty jednotlivých vrstiev. V prípade že ťahy pochádzali z hier, ktoré skončili prehrou, znamienka gradientov boli otočené. Pri spätnom priechode boli teda pričítané opačné gradienty. Experimenty však ukázali, že táto metóda nie je vhodná. Strategická sieť pri hraní nedokázala určiť zmysluplné ťahy.

7.6 Rôzne konvolučné jadrá strategickej siete

Ďalším experimentom bolo trénovanie strategických sietí s tromi konvolučnými vrstvami a rôznymi veľkosťami konvolučných jadier. Účelom bolo zistiť, či je lepšie začať menším alebo väčším konvolučným jadrom a do akej miery pomáha zväčšovanie okolia, o ktorom má každé políčko na hracej ploche informácie. Všetky neurónové siete majú tri konvolučné vrstvy a každá, okrem poslednej, je nasledovaná ReLU vrstvou a vrstvou pre normalizáciu dávky.

Validácia na testovacej dátovej sade, zaznamenaná na obrázku 7.5, ukázala, že zväčšovanie okolia zvyšovalo presnosť. Toto ale zrejme neplatilo pri štvrtej architektúre, kde bola presnosť v oboch meraniach o niečo menšia. Zrejme nie je vhodné, aby konvolučné jadro poslednej vrstvy bolo príliš veľké vzhľadom na konvolučné jadrá predošlých vrstiev.

7.7 Strategická sieť s prehľadávacím algoritmum

Nasledujúcim algoritmom bolo hranie vytvorenej umelej inteligencie s použitím prehľadávacieho algoritmu popísaného v kapitole 6.2. Pri prehľadávaní bolo vybraných 5 ťahov s najväčšou pravdepodobnosťou, z ktorých bolo pomocou Strategickej siete generovaných 32 hier.

Pozorovaním priebehu hier sa zistilo, že umelá inteligencia hrá príliš ofenzívne a niekoľkokrát si nevšimla protihráčovu trojicu alebo štvoricu žetónov, ktorú by ešte mohla zablokovať. Umelá inteligencia sa väčšinou snažila vytvoriť čo najdlhší rad vlastných žetónov. Proti programu Imro nedokázala vyhrať. Napriek tomu že umelá inteligencia občas vybrala správny ťah, nebolo to na výhru dostačujúce. K výhre by musela byť zaručená vyššia konzistentnosť pri výbere vhodného ťahu.

Nedostatočná konzistentnosť mohla byť taktiež spôsobená malým počtom generovaných hier. Problémom bolo že stroj, na ktorom boli experimenty vyhodnocované, nemal dostatočne výkonný grafický procesor na to, aby pri vyššom počte generovaných hier dokázal určiť nasledujúci ťah v rozumnom čase.

7.8 Ohodnocovacia sieť

Posledným experimentom bolo natrénovanie ohodnocovacej siete a jej použitie v druhom algoritme z kapitoly 6.2. Mnou použité architektúry, ktoré boli podobné architektúre na obrázku 6.2, po natrénovaní neurčovali zmysluplné výsledky pri verifikácii. Pri pokuse o určenie pravdepodobnosti v situácii, keď jeden z hráčov mal na hracej ploche položenú výhernú kombináciu žetónov, pravdepodobnosť výhry určená neurónovou sieťou sa nepribližovala reálnej pravdepodobnosti. V budúcnosti plánujem vytvoriť vhodnú architektúru a takúto neurónovú sieť natrénovať správnym spôsobom tak, aby určovala zmysluplné pravdepodobnosti.

Kapitola 8

Záver

Táto práca sa zaoberala vytvorením umelej inteligencie pre hru Gomoku. Pre určenie nasledujúceho ťahu boli navrhnuté dve neurónové siete a dva algoritmy. Narozdiel on bežných programov, táto umelá inteligencia nepoužíva žiadne vopred definované pravidlá alebo heuristiky pre určenie ťahu. Strategická neuronová sieť určuje pravdepodobnostnú distribúciu na základe konfigurácie hracej plochy. Ohodnocovacia neurónová sieť určuje výsledok hry pomocou hracej plochy. Pre trénovanie je použitá sada hier vytvorená hraním programov zúčastnených v súťaži Gomocup. Týmto postupom sa podarilo vytvoriť trénovaciu a testovaciu sadu s približne 17000 unikátnymi hrami. Natrénovaná strategická sieť je schopná konkurovať nižšie umiestneným programom v súťaži Gomocup. V budúcnosti je plánované vytvoriť vhodnú architektúru pre ohodnocovaciu sieť a neurónovú sieť natrénovať ju tak, aby určovala zmysluplné pravdepodobnosti výhry daných hier.

Literatúra

[1] Ford, D.: Gomoku AI Player. Technická zpráva, Cardiff University, Květen 2016.

[2] Gatys, L. A.; Ecker, A. S.; Bethge, M.: A Neural Algorithm of Artistic Style. *CoRR*, ročník abs/1508.06576, 2015.

[3] Ioffe, S.; Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, ročník abs/1502.03167, 2015.

[4] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *CoRR*, ročník abs/1408.5093, 2014.

[5] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, 2012, s. 1097–1105.

[6] Lecun, Y.; Bottou, L.; Bengio, Y.; aj.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, ročník 86, č. 11, Nov 1998: s. 2278–2324.

[7] Liu, W.; Wen, Y.; Yu, Z.; aj.: Large-Margin Softmax Loss for Convolutional Neural Networks. *ArXiv e-prints*, Prosinec 2016.

[8] Schaul, T.; Schmidhuber, J.: A scalable neural network architecture for board games. In *2008 IEEE Symposium On Computational Intelligence and Games*, dec 2008.

[9] Silver, D.; aj.: Mastering the game of Go with deep neural networks and tree search. *Nature*, ročník 529, č. 7587, Jan 2016: s. 484–489, article.

[10] Zhang, R.: Convolutional and Recurrent Neural Network for Gomoku. Technická zpráva, Stanford University, 2016.

[11] Šíma, J.; Neruda, R.: *Teoretické otázky neuronových sítí*. Matfyzpress, první vydání, 1996.

Prílohy

Príloha A

Obsah CD

Pribalené CD obsahuje:

AI Adresár obsahujúci natrénované neurónové siete, skripty pre hranie a trénovanie neurónových sietí a iné použité zdrojové kódy.

preview Adresár obsahujúci prezentáciu a z nej vyexportované video.

thesis Adresár s bakalárskou prácou a zdrojovými kódmi v LaTeXe.