Neurónové siete a AlphaGo

Dominika Kubániová

Abstrakt

Medzi najnovšie prelomové udalosti v oblasti umelej inteligencie patrí výhra počítača AlphaGo v strategickej hre Go nad profesionálnym hráčom Lee Se-Dolom. Svoju prácu som sa rozhodla venovať práve tomuto stroju a jeho vývoju. Vypracovanie AlphaGo ku stroju, ktorý je schopný ukázať takýto výkon, spočíva vo využití neurónových sietí, a ich schopnosti učiť sa, s kombináciou vyhľadávacích stromových algoritmov. Preto vo svojej práci najprv ujasním pojmy a matematické reprezentácie, ktoré sú potrebné pre vysvetlenie stavby a fungovania počítača AlphaGo, a neskôr ukážem ako vývojári z projektu Deep Mind dokázali vytvoriť tak silný stroj, ktorého rozhodnutia pripomínajú rozhodnutia ľudského hráča vytvorené intuíciou a emóciami.

1 Definícia neurónu a neurónovej siete

Umelé neurónové siete boli inšpirované neurónovou sústavou v našom ľudskom tele. Ich stavba a prvky, z ktorých pozostávajú sú analogické väčšine zakladným biologickým funkciám neurónu.

Neurón je základná stavebná jednotka umelej neurónovej siete, ktorý má ľubovolný počet vstupov, nazývaných synapsie, a jeden výstup, ktorý je reprezentovaný jednou hodnotou, šírenou ďalej neurónovou sieťou. Čím viac synapsií, teda spojníc medzi neurónami, sieť obsahuje, tým má možnosť uschovať väčšie množstvo dát. *Neurónovou sieťou* rozumieme rozloženie jednotlivých neurónov vo vrstvách.

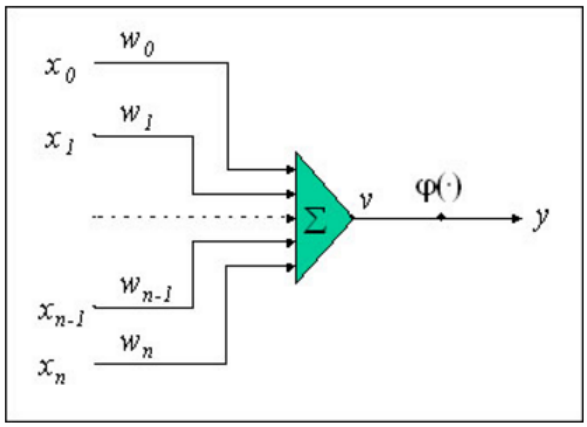
2 Matematický model neurónu

Matematicky môžeme neurón definovať ako funkciu $f : \mathbb{R}^{N+1} \times \mathbb{R}^{N+1} \to \mathbb{R}$ pričom

$$f : (\mathbf{x}, \mathbf{w}) \mapsto f(\mathbf{x}, \mathbf{w}) = \varphi(v)$$

kde dvojica $(\mathbf{x}, \mathbf{w})$ je vstupný vektor $\mathbf{x} = (x_0, ..., x_N)^T \in \mathbb{R}^{N+1}$ ohodnotený váhovým vektorom $\mathbf{w} = (w_0, ..., w_N)^T \in \mathbb{R}^{N+1}$; $\varphi : \mathbb{R} \to \mathbb{R}$ aktivačná funkcia a $v \in \mathbb{R}$ vnútorný neurónový potenciál získaný štandardným skalárnym súčinom $(\mathbf{x} \cdot \mathbf{w})$, teda $v = \sum_{i=0}^{N} x_i w_i$. Váhový vektor, inak nazývaný ako váha neurónu, je parameter, ktorého veľkosť vyjadruje „skúsenosť“ neurónu, teda čím je táto

hodnota vyššia, tým je daný vstup dôležitejší. Váhy sú väčšinou inicializované na malé náhodné hodnoty v intervale $\langle -\delta, \delta \rangle$, kde $\delta \in (0, 1)$ [24].



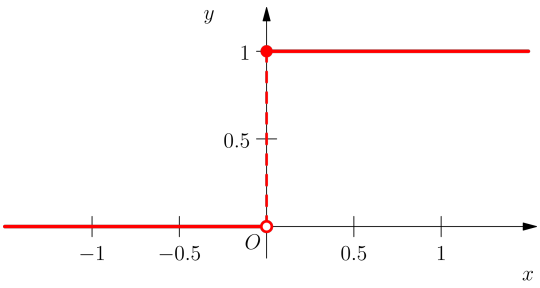
Obr. 1: Znázornenie matematického modelu a matematický zápis [1]

$$y = f(\mathbf{x}, \mathbf{w}) = \varphi(\sum_{i=0}^{N} x_i w_i)$$

Aktivačná funkcia $\varphi(\cdot)$ umožňuje vykonávať komplexnejšie výpočty, pričom výstup upravuje na hodnoty, ktoré sú ďalej šírené sieťou.

Najjednoduššou aktivačnou funkciou je **skoková funkcia** *(obr. 2)*, ktorá vracia hodnotu

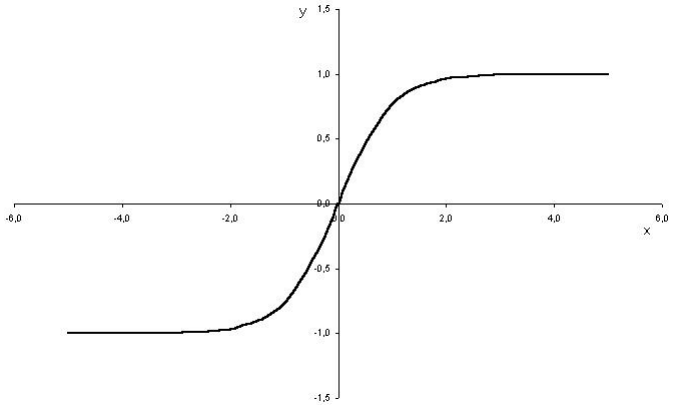
$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & inak \end{cases}$$



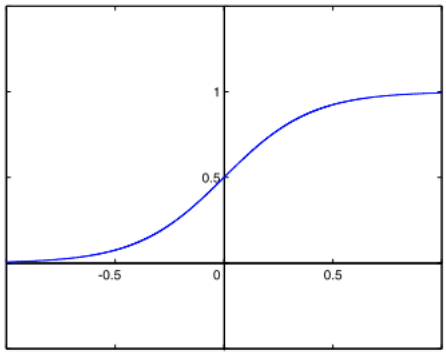
Obr. 2: Skoková funkcia [20]

Ďalšími základnými aktivačnými funkciami sú **hyperbolický tangens** *(obr. 3)*, ktorý je definovaný pre $\forall v \in \mathbb{R}$ a vracia hodnoty z intervalu $(-1, 1)$; a

najpoužívanejšia **sigmoida** *(obr. 4)* definovaná pre $\forall v \in \mathbb{R}$ s oborom hodnôt $(0, 1)$.



Obr. 3: Hyperbolický tangens [21]



Obr. 4: Sigmoida [22]

Príklad neurónovej siete: Neurónová sieť má dve vrstvy *(obr. 5)*. Nech f_1, f_2, f_3 sú neuróny v prvej vrstve a g neurón v druhej vrstve. Nech $\mathbf{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3$ je vstupný vektor pre neuróny f_1, f_2, f_3; $\mathbf{a} = (a_1, a_2, a_3) \in \mathbb{R}^3$ výstupný vektor pre neuróny f_1, f_2, f_3 a vstupný vektor pre neurón g; a $y \in \mathbb{R}$ je výstup neurónu g a celej neurónovej siete. Nech $W = (w_{ij}^{(1)}) \in \mathbb{R}^{3\times 3}$ je matica váhových vektorov kde j-ty stĺpec je váhový vektor f_j -teho neurónu a $\mathbf{w}^{(2)} = (w_1^{(2)}, w_2^{(2)}, w_3^{(2)})$ je váhový vektor neurónu g v druhej vrstve. Ďalej nech $\varphi : \mathbb{R} \to \mathbb{R}$ je aktivačná funkcia. [9]

Podľa vzťahu uvedenom v matematickom modely sa a_1, a_2, a_3 rovnajú

$$a_1 = \varphi(x_1 w_{11}^{(1)} + x_2 w_{21}^{(1)} + x_3 w_{31}^{(1)})$$
$$a_2 = \varphi(x_1 w_{12}^{(1)} + x_2 w_{22}^{(1)} + x_3 w_{32}^{(1)})$$

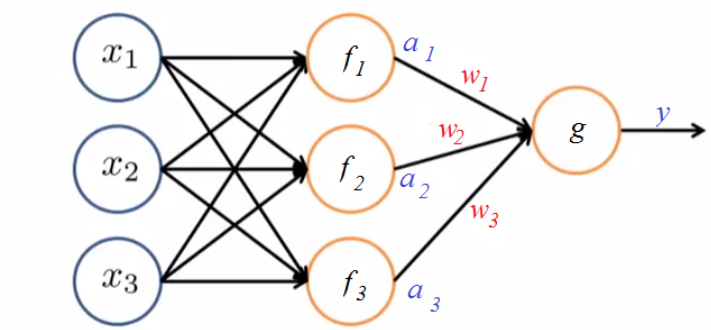
$$a_3 = \varphi(x_1 w_{13}^{(1)} + x_2 w_{23}^{(1)} + x_3 w_{33}^{(1)})$$

Potom

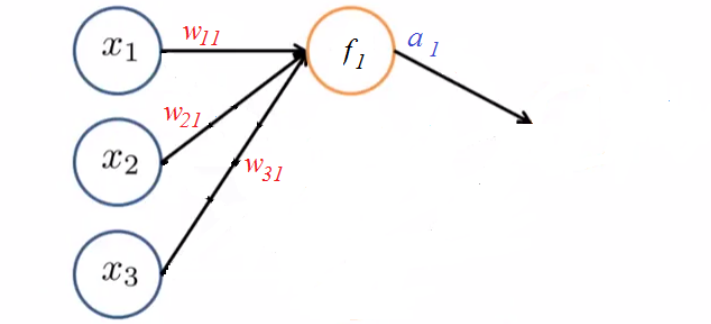
$$y = \varphi(a_1 w_1^{(2)} + a_2 w_2^{(2)} + a_3 w_3^{(2)})$$

alebo inak zapísané

$$y = \varphi(\mathbf{a} \cdot \mathbf{w}^{(2)})$$



Obr. 5: Príklad neurónovej siete (hore) a detail váhového vektoru neurónu f_1 nachádzajúceho sa v prvom stĺpci matice W (dole)



3 Učenie

Zaujímavou vlastnosťou neurónových sietí je ich schopnosť učiť sa. Strojové učenie, známejšie pod názvom „machine learning“, Arthur Samuel výstižne definoval ako „Odbor, ktorý dáva počítačom schopnosť učiť sa, bez toho, aby boli explicitne naprogramované.“ [7]. Neurónové siete učiňujú proces učenia prispôsobovaním si váh synapsií medzi neurónmi v sieti. Posilnením, resp. zoslabením týchto spojníc umožňujú sieti možnosť učiť sa. Učiace algoritmy vlastne pomáhajú v situáciách kedy problém je tak zložitý, že nedokáže byť človekom naprogramovateľný.

Algoritmy strojového učenia rozdelené podľa spôsobu učenia, ako fungujú a ich použitie:

Učenie s učiteľom *(supervised learning)*
Počítač dostane sadu dvojíc vstupov a výstupov *(x,y)*, kde vstup $x \in X$ a výstup $y \in Y$. Nech F je trieda funkcií. Cieľom je nájsť funkciu $f \in F$, $f : X \to Y$, ktorá spája konkrétny vstup x s jeho správnym výstupom y [2]. Algoritmus sa učí porovnávaním aktuálneho výstupu so správnym výstupom k x, vzhľadom k čomu následne upraví váhy v neurónovej sieti. Prostredníctvom metód ako sú napr. regresia, klasifikácia alebo predikácia, učenie s učiteľom využíva tento vytvorený vzor pre predpovedanie hodnôt na ďalších nespracovaných dátach. Učenie s učiteľom sa používa v alikáciach pre rozpoznávanie reči či obrazov; v medicíne pri detekcií nádorov.

Učenie bez učiteľa *(unsupervised learning)*
V tomto type učení počítač dostane iba sériu vstupov, teda povedzme že iba množinu X. Algoritmus musí sám bez pomoci prísť na to, aké dáta mu ukazujeme. Cieľom je tieto dáta preskúmať a nájsť v nich nejaké štruktúry. Využitie môžeme nájsť napr. v reklamách, ktoré sa nám ukazujú na okrajoch internetových stránok. Algoritmus si zozbiera informácie o tom čo často vyhľadávame alebo nakupujeme, a podľa toho nám odporúča nové “produkty, ktoré by nás mohli zaujímať“ na rôznych stránkach, ktoré dokonca s produktmi nemusia mať nič spoločné.

Posilované učenie *(reinforcement learning)*
Počítač zvyčajne nedostane ani vstupné dáta. Trénovanie algoritmu spočíva v pozorovaní vonkajšieho prostredia a hľadaní akcií prostredníctvom pokusov a omylov, ktoré prinesú najväčšiu “odmenu“. Tento typ učenia pozostáva z troch dôležitých kompononetov:
1. *Agent (teda stroj, ktorý sa učí)*
2. *Vonkajšie prostredie (ktoré agent pozoruje; prostredie môže byť aj virtuálne)*
3. *Akcie (čo sa agent učí robiť)* [8]
S posilovaným učením sa stretneme hlavne v robotike a v hrách.

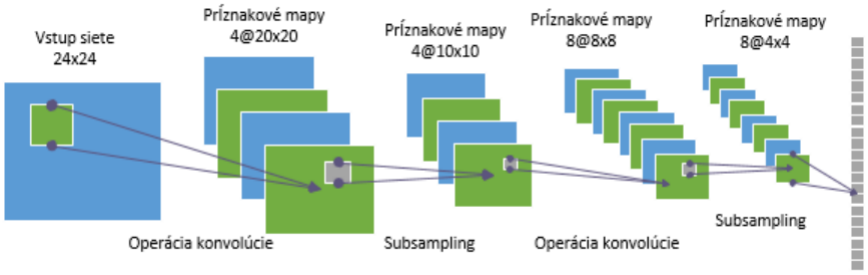
4 Konvolučné neurónové siete

Konvolučné siete sú typ umelých neurónových sietí, ktoré boli navrhnuté pre rozpoznávanie dvojrozmerných obrazových vzorov priamo z pixelov vstupných obrázkov [17]. Z tohoto dôvodu ich uplatnenie nájdeme hlavne v rozpoznávaní objektov alebo tvárí vo vstupnom obraze. Konvolučné siete pozostávajú z konvolučných vrstiev a subsamplingových vrstiev, ktoré sa navzájom striedajú. Na konci sa nachádza výstup poslednej subsamlingovej vrstvy transformovaný do jednorozmerného vektoru. Každý neurón v konvolučnej vrstve je prepojený s istým malým okolím (recepčným polom) spracovávaného obrázku, napr. jedným pixelom, vďaka čomu získava základné príznaky, ako sú hrany, rohy, farby atď.

Na vstupný obrázok je v prvej konvolučnej vrstve aplikovaný filter, ktorého cieľom je získať príznaky obrázku. Recepčné pole sa aplikuje na každý pixel, čím sa počíta jedna operácia na rôznych častiach obrázka. V konvolučnej vrstve všetky neuróny zdieľajú tú istú množinu váh, čo obmedzuje celkový počet výpočtov a tým zrýchluje celý proces. Množina výstupov neurónov v konvolučnej vrstve sa nazýva príznaková mapa.

Nasleduje subsamplingová vrstva, ktorej úlohou je zredukovať veľkosť vstupu, teda veľkosť príznakových máp. Najpoužívanejšou funkciou v subsamplingovej vrstve je vyberanie maxima. Z každej príznakovej mapy si funkcia vyčlení istú oblasť, povedzme že o rozmere $k \times k$, z predchádzajúcej vrstvy, povedzme že o rozmere $n \times n$, a z nej ako výstup vyberie maximálnu hodnotu z tejto oblasti. Tento výstup bude o rozmere $\frac{n}{k} \times \frac{n}{k}$.

Po niekoľkých vystriedaniach konvolučných a subsamplingových vrstiev je výstup transformovaný do jednorozmerného vektoru, na ktorý je napojená klasická neurónová sieť, ktorá vypočíta konečný výstup *(obr. 6)* [18].



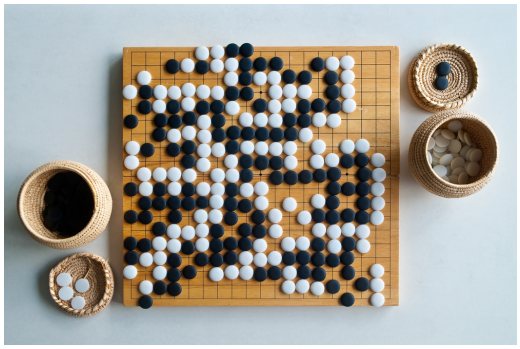
Obr. 6: Príklad architektúry konvolučnej neurónovej siete [18]

5 AlphaGo

V roku 1997 bol Garry Kasparov, majster v hraní šachu, porazený v súboji s počítačom Deep Blue. Počítač mal vtedy ale vopred naprogramovaný "manuál", ako má hru hrať tak, aby vyhral. Tento rok vývojári umelej inteligencie z pro-

jektu Deep Mind vyzvali na súboj druhého najlepšieho hráča na svete v hre Go, Lee Se-Dola, proti ich počítaču s názvom AlphaGo. Všetkých počítač ohromil, keď nejeden raz donútil Lee Se-Dola vzdať sa. Narozdiel od počítaču Deep Blue, AlphaGo na začiatku svojej existencie poznal iba pár elementárnych pravidiel ako hru Go hrať, ktoré ani zďaleka nestačili k tomu, aby dokázal vyhrať nad akýmkoľvek amatérskym hráčom. Cieľom Deep Mind nebolo naprogramovať počítač, ktorý vie hru hrať ako profesionálni hráči od jeho počiatku, ale vytvoriť počítač, ktorý sa všetky svoje schopnosti dokáže naučiť sám.

Najprv si ale povieme v čom hra Go spočíva a prečo je výhra AlphaGo nad Lee Se-Dolom tak prelomová v oblasti umelej inteligencie. Go je strategická hra pre dvoch hráčov, ktorá spočíva hlavne v intuícii a inštinkte hráča. Jej pravidlá su veľmi jednoduché, ale hra vyžaduje veľké logické myslenie, trpezlivosť, dôvtipnosť, rozhodnosť a schopnosť vcítiť sa do uvažovania súpera. Presne takéto ľudksé emócie by sme v počítači asi nehľadali. Go sa hrá na doske s mriežkou o rozmere 19x19, pričom hráčí nastriedačku pokladajú čierne a biele kamienky na priesečníky mriežky. Cieľom je ohraničiť čo najväčšie územie na doske a zajať súperové kamienky. Na konci hry si obidvaja hráči spočítajú všetky priesečníky územia a počet zajatcov. Hráč s väčším počtom vyhráva [11].



Obr. 7: hra Go [23]

Na väčšinu strategických hier môžeme aplikovať rekurzívny stromový vyhľadávací algoritmus *minimax*, ktorý dokáže vyhodnotiť výsledok hry z každého jej stavu s. Ohodnocovaciou funkciou $v^*(s)$ algoritmus určí k akému výsledku vedie každý jeden ťah, za predpokladu že hráči vedú perfektnú hru (teda že si vyberajú iba tie ťahy, ktoré im bezprostredne prinesú výhru). Následne algoritmus vyberie len tie najpriaznivejšie ťahy, ktoré vedú k výhre.

Počet možných ťahov v hre Go sa odhaduje na približne 10^{800}, čo je rozhodne viac ako je atómov vo vesmíre (okolo 10^{80}). Kvôli takémuto astronomickému množstvu možných ťahov je využitie minimaxu, hrubej sily alebo heuréky, prípadne kombinácie všetkých troch ako to bolo v počítači Deep Blue, zďaleka nereálne.

5.1 Učenie AlphaGo

Doposiaľ všetky stroje, ktoré hrali strategické hry sa riadili vyhľadávacími stromami a vopred človekom naprogramovanými pravidlami. Aby sa týmto pravidlám AlphaGo vyhlo, tvorcovia pri jeho vývoji uplatnili strojové učenie. AlphaGo kombinuje konvolučné siete a algoritmus vyhľadávacieho stromu Monte Carlo. Vyhľadávací strom Monte Carlo je stromový algoritmus, ktorý vyhodnocuje každé štádium hry na základe priemerných výsledkov simulácií. Simulácia začína v aktuálnom štádiu hry a končí vtedy, keď jeden z hráčov vyhrá. Narozdiel od minimaxu nezvažuje všetky možné ťahy, ale vyberá si len tie najprínosnejšie. V každom uzle stromu sú uchovávané dáta typu ako často bol uzol navštívený, aký ťah sa v ňom vykonáva a aká je jeho pravdepodobnosť, že vedie k výhre. Najprv algoritmus vykonáva náhodné simulácie počas ktorých zozbiera tieto dáta. Čím menej sú simulácie náhodné, tým viac sa presnosť dát zvyšuje. Nakoniec si algoritmus vyberie tú najvhodnejšiu cestu ku výhre.

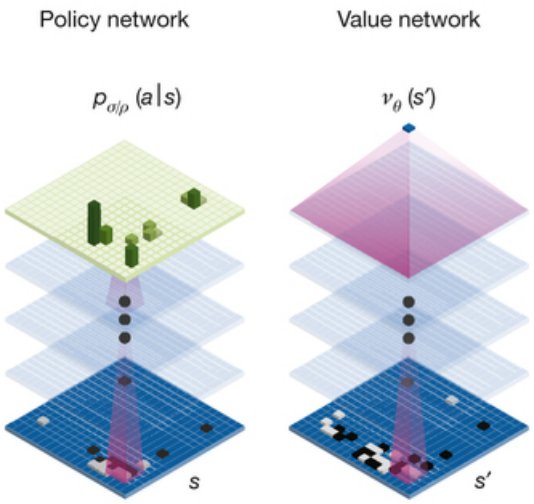
AlphaGo využíva tri typy konvolučných sietí: jednu vyhodnocovaciu sieť v *(value network)*, ktorá vyhodnocuje víťaza hry, taktickú sieť p *(policy network)*, pre predpovedanie a vyberanie ďalších ťahov, a simulačnú sieť *(rollout network)*, pre zrýchlenie procesu. Vstupom pre všetky typy sietí je obraz aktuálneho štádia a rozloženia kamienkov na doske. Prvé vrstvy konvolučných sietí obsahujú filtre, ktoré spracovávajú vstupný obraz, rozpoznávaním tvarov a farieb.

Samotné učenie počítaču AlphaGo spočívalo z niekoľkých etáp. Prvou bolo **učenie s učiteľom** taktickej siete p_σ, aby bola schopná vykonávať ťahy, ktoré sa najviac podobajú tým ľudským za akéhokoľvek štádia na hracej doske. To dosiahli ukázaním počítaču 30 miliónov ťahov profesionálnych hráčov, teda ukázaním dvojíc $(s|a)$ kde s je vstup reprezentovaný aktuálnym rozostavením kamienkov na hracej doske, ktorý prejde konvolučnými sieťami s váhami σ. Posledná výstupová vrstva siete vracia rozdelenie pravdepodobnosti $p_\sigma(s|a)$ nad všetkými prístupnými ťahmi a *(obr.8)*. 13-vrstvové konvolučné siete sa nezaujímali o výhru, ale len o vyberanie ťahov, ktoré by si vybrali tí najsilnejší hráči. Slabosť počítača v tomto štádiu učenia ale spočívala v tom, že pri hraní v podstate imitoval profesionálných hráčov. AlphaGo bol síce dobrý v predpovedaní ťahov súpera (jeho výber sa zhodoval s výberom hráčov 57% času), ale cieľom bolo zdokonaliť sieť tak, aby hru vyhrala.

Tento ciel dosiahla druhá etapa, ktorou bolo **posilované učenie** taktických sietí p_ρ s rovnakou stavbou ako siete učené s učteľom, s váhami ρ takými že $\rho = \sigma$. Posilované učenie spočívalo v nechaní taktických sietí p_ρ hrať tisíce hier proti predchádzajúcim sieťam učených s učiteľom p_σ, čím sa naučili objavovať sami nové stratégie. Inak povedané, nechali počítač hrať samého so sebou. Tieto taktické siete boli trénované, aby vyberali práve tie výherné ťahy. Výsledkom boli také silné taktické siete, že vedeli v okamihu, bez akéhokoľvek stromového vyhľadávania, poraziť programy, ktoré stavali veľké vyhľadávacie stromy reprezentujúce hru.

Poslednou etapou bolo trénovanie vyhodnocovacej siete na 30-tich miliónoch herných pozíciách získaných počas hrania taktických sietí samích proti sebe. Ku každej hre poznali jej výsledok, preto boli vývojári schopní vytvoriť vyhodno-

covaciu funkciu P a na týchto dátach naučiť vyhodnocovaciu sieť v_θ s váhami θ predpovedať konečného víťaza $v_\theta(s')$ v štádiu s' po ťahu a. *(obr. 8)*



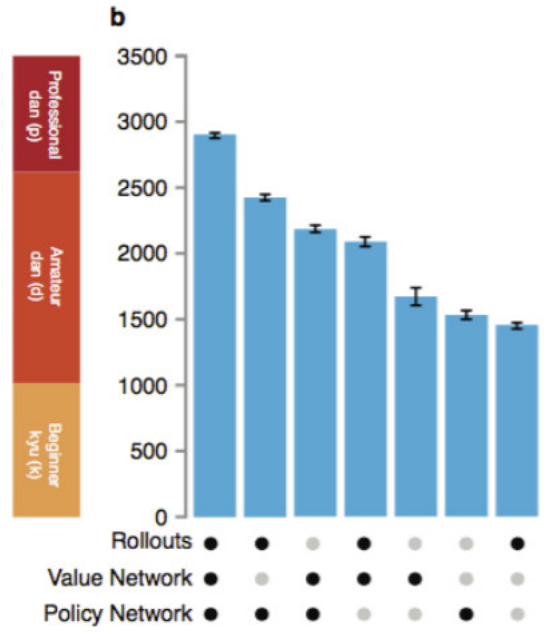
Obr. 8: Znázornenie architektúry taktickej a vyhodnocovacej siete. Výstupom taktickej siete je rozdelenie pravdepodobnosti medzi možné ťahy. Veľkosť zelených tyčiek na vrchnej zelenej doske na obrázku reprezentuje pravdepodobnosť výhry ťahu. Výstupom vyhodnocovacej siete je jediné reálne číslo medzi 0 a 1. Výhra bieleho hráča je indikovaná číslom 0 a výhra čierneho hráča 1. Číslo $0,5$ znamená remízu. [13]

Napriek tomu, že samotná taktická sieť podávala výborné výsledky, jej výber nových ťahov bol dosť pomalý, keďže pre vyhodnotenie správnej odpovede potreboval algoritmus prejsť všetkými tisícami možnými ťahmi. Vyššiu rýchlosť vyriešila nová simulačná sieť, ktorá sa nepozerala na celú hraciu dosku, ale len na menšie okolie súperovho predchádzajúceho ťahu. Táto metóda len zanedbateľne zoslabila schopnosť taktických sietí správne predpovedať, ale výrazne urýchlila celý proces (ako možno vidieť v grafe *obr. 9*). [13]

5.2 Kombinácia konvolučných sietí s vyhľadávacím stromom

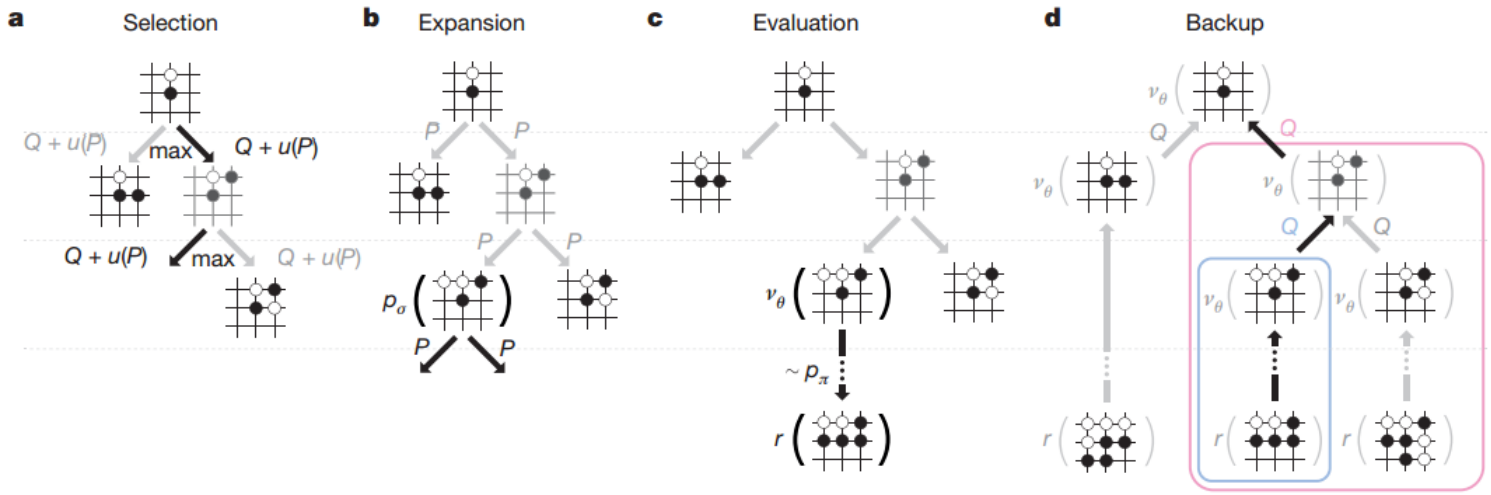
Samotné využitie algoritmu vyhľadávacieho stromu Monte Carlo by bolo neefektívne kvôli dvom problémom, ktorými boli obrovská šírka a hĺbka stromu. Vedením vyhľadávania v strome tromi typmi konvolučných sietí tieto problémy vyriešilo.

(Následujúci algoritmus je znázornený na *obr. 10*) Nech s je súčasný stav hry a a akcia, tedá následujúci ťah hráča. Každý uzol stromu reprezentuje istú akciu a obsahuje dáta: hodnotu akcie $Q(s|a)$ a výstup vyhodnocovacej siete $P(s|a)$.



Obr. 9: Výkon AlphaGo pri rôznych kombináciach jeho sietí. Vertikálna osa reprezentuje systém Elo rating. [13]

Chceme docieliť, aby algoritmus postupoval cestou s najväčšími hodnotami Q. Predstavme si, že algoritmus je v stave rozhodovania, už má nejaké vyhľadávanie v strome za sebou a narazil na uzol (**a.** Selection, posledná čierna šípka do ľava), ktorý ešte nebol spracovaný ani vyhodnotený. Na neho sa zavolajú taktické siete p, ktoré strom z tohoto stavu expandujú (**b.** Expansion). Namiesto pridania všetkých možných ťahov, taktické siete obmedzia stovky nových ťahov na 3-4 najvýhodnejšie, čím redukujú šírku stromu. Následne je zavolaná vyhodnocovacia sieť v, ktorá vyhodnotí P všetky nové ťahy, teda či vedú ku výhre, prehre alebo remíze. Tento výsledok je kombinovaný so štatistickou analýzou simulačnej siete, ktorá spustí simuláciu hry až do jej konca (**c.** Evaluation), počas ktorej zozbiera dáta (počet výhier a prehier oboch hráčov). Na konci simulácie funkcia $r(\cdot)$ vyhodnotí víťaza hry. Toto vedie k zníženiu hĺbky stromu. Poslednou etapou je vrátenie sa zo simulácií ku koreňu stromu, teda ku východziemu stavu, počas ktorého sa na ceste hore aktualizujú hodnoty Q na základe výsledkov z vyhodnocovacej siete v a funkcie $r(\cdot)$. AlphaGo konečne vyberie akciu s najvyššou hodnotou Q. [13]



Obr. 10: Znázornenie vyhľadávania v strome [13]

5.3 Výsledky

Nakoniec spomeniem zopár úspechov, ktoré AlphaGo dosiahol. Ako prvé boli testované jeho taktické siete proti dovtedy najsilnejšiemu hraciemu stroju Patchi, ktorý sa výhradne spolieha na 100 000 simulácií vyhľadávacieho stromu Monte Carlo v každom ťahu. AlphaGo vyhral 85% odohraných zápasov. Systém, ktorý hodnotí silu hráčov Go sa nazýva Elo rating, ďalej len Elo. Prvým ľudským profesionálnym hráčom, s ktorým AlphaGo hral, bol Fan Hui (Elo 2908). AlphaGo malo v tom čase Elo len 2890, ale aj napriek tomu dokázalo vyhrať všetky súboje. Ďalší turnaj viedol počítač proti Lee Se-Dolovi, druhému najlepšiemu hráčovi Go na svete s Elo 3520. V tej dobe malo AlphaGo Elo 3140, ktoré stačilo na to, aby Lee Se-Dola porazil v štyroch hrách z piatich. Takýto veľkolepý úspech mu zvýšil Elo na 3586. Jediným existujúcim človekom na svete s vyšším Elo ako AlphaGo je už len Ke Jie s 3621. Aj keď sa tento súboj ešte neuskotočnil, na základe získaných poznatkov o stroji AlphaGo môžeme ľahko predpovedať ako by zrejme dopadol [12].

Literatúra

[1] https://software.intel.com/en-us/articles/an-introduction-to-neural-networks-with-an-application-to-games

[2] https://en.wikipedia.org/wiki/Artificial_neural_network

[3] http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-2-learning/8

[4] V.Kvasnička, Ľ. Beňušková, J.Pospíchal, I.Farkaš, P.Tiňo, A.Kráľ: **Úvod do teórie neurónových sietí**

[5] Mgr. Lubomír Šerý: **Metody umělé inteligence a jejich využití při predikci** (Diplomová práca)

[6] https://is.cuni.cz/webapps/zzp/detail/63627/

[7] https://en.wikipedia.org/wiki/Machine_learning

[8] http://www.sas.com/en_id/insights/analytics/machine-learning.html

[9] Stanford University: **Machine Learning**, Andrew Ng (online kurz), 6. prednáška, čas: 29:00

[10] http://togelius.blogspot.sk/2016/01/why-video-games-are-essential-for.html

[11] https://sk.wikipedia.org/wiki/Go

[12] https://www.tastehit.com/blog/google-deepmind-alphago-how-it-works/

[13] David Silver a kol.: **Mastering the game of Go with deep neural networks and tree search** (článok)

[14] https://www.youtube.com/watch?v=f71RwCksAmI&list=WL&index=10

[15] https://www.dcine.com/2016/01/28/alphago/

[16] https://en.wikipedia.org/wiki/Convolutional_neural_network

[17] Mgr. Matej Hrinčár: **Konvoluční neuronové sítě a jejich využití při detekci objektů** (Diplomová práca), Kapitola 3

[18] https://core.ac.uk/download/pdf/30299309.pdf

[19] http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

[20] https://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/Heaviside.svg/220px-Heaviside.svg.png

[21] https://upload.wikimedia.org/wikipedia/commons/1/14/TanhFunction.jpg

[22] https://upload.wikimedia.org/wikipedia/commons/9/96/SigmoidFunction.svg

[23] http://blogs.discovermagazine.com/crux/files/2016/01/shutterstock_342026228.jpg

[24] Sartaj Singh Sodhia, Pravin Chandra: **Interval Based Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networks**