Användningen av maskininlärning i artificiell intelligens till strategi baserade spel

Linus Wiberg 42101
Kandidatavhandling i datavetenskap
Handledare: Luigia Petre
Fakulteten för naturvetenskap och teknik
Åbo Akademi
31 mars 2020

Innehåll

Referat

Datorspelare som funktionerar som motståndare i spel gör det möjligt att spela spel om man inte har en mänsklig motståndare. Datorspelare i de flesta strategispel är ännu i dagens läge svaga. I denna avhandling undersöks användningen av olika sorters maskininlärning i artificiell intelligens som spelar strategi baserade spel.

Sökord: Maskininlärning, artificiell intelligens, strategispel

1. Inledning

De flesta av dagens spel använder någon typ av Artificiell Intelligens (AI) för att styra non-player-characters (NPC), spelfigurer som inte styrs av en annan riktig spelare, eller bottar så att de ska kunna reagera på ett trovärdigt och kompetent sätt när en situation uppstår i ett spel. Även om dagens spel-AI kallas AI och marknadsförs som sådan, så har de oftast bara hårdkodad respons för olika situationer. Detta gör att de ofta är extremt begränsade i sina beslut.

I denna avhandling undersöks AI i formen av motspelare, specifikt i strategi baserade-spel, som använder sig av maskininlärning. AI i strategispel kan dra stor nytta av förmågan av att fatta kloka och snabba beslut, i andra genrers datorspel kan dator-spelare vara dugliga utan att göra kloka beslut eftersom de inte har mänskliga nackdelar såsom långsamma reflexer eller dålig syn.

Vissa datorspelare i strategispel som har svårighetsgrader fuskar vid högre svårighet istället för att göra kompetenta beslut. Detta gör de för att de inte är tillräckligt bra. I ett flertal strategispel där man använder resurser för att skapa trupper och byggnader så fuskar datorspelaren genom att få mera resurser än en riktig spelare för samma gärningar. Detta är inte optimalt för det kan få den verkan att spelaren inte har roligt när den spelar emot en datorspelare eftersom den antingen fuskar eller är extremt dålig.

I digitala brädspel såsom schack har det länge experimenterats med att skapa motståndarspelare som kan vinna mot människor. Schackmotorer som dessa kallas har varit bättre än de flesta mänskliga spelarna länge, men även dessa har dock blivit utklassade av artificiell intelligens som använder sig av maskininlärning.

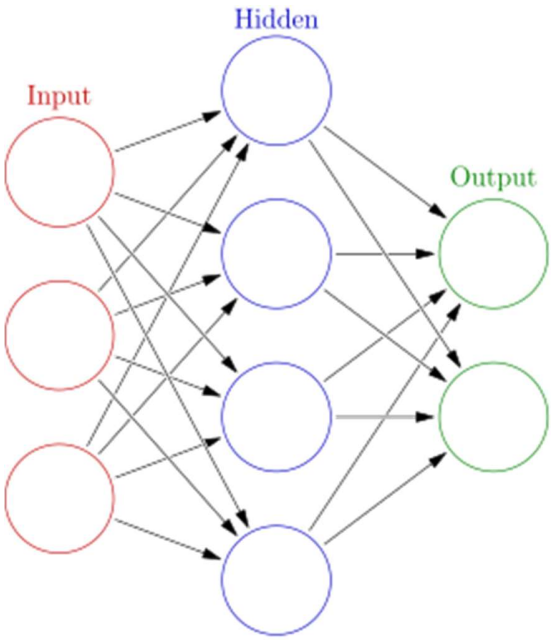
2. Termer inom avhandlingen

Förklaringar på de termer som används i denna avhandling. Några termer kan ha en annan betydelse i andra kontext.

2.1 Artificiellt Neuronnät

Artificiella neuronnät (ANN), på engelska artificial neural network, är de självlärande algoritmerna som försöker imitera eller är inspirerade av biologiska neuronnät som finns i hjärnor[1]. Neuronnät är uppbyggda av flera lager sammankopplade artificiella neuroner som kan överföra signaler mellan varandra[1].

Det första lagret neuroner, i figur 2.1 det röda lagret, är inmatnings lagret, i inmatning lagret kan det finnas flera artificiella neuroner men bara ett lager neuroner. Inmatnings lagret är kopplat till de dolda lagren, i figur 2.1 det blåa lagret, de dolda lagren, som det kan finnas flera lager av, använder indatan som härstammar från inmatnings lagret för att producera data som den skickar till utmatnings lagret[1], i figur 2.1 det gröna lagret. När datan har kommit igenom neuronnätet till utmatnings lagret skickar neuronnätverket ut informationen, exempelvis till en kontroll system[1].



Figur 2.1. En figur på ett simpelt neuronnätverk.

Varje koppling mellan neuronerna har en vikt, när en neuron får indata, i formen av siffror, multiplicerar den siffrorna med vikten på den kopplingen som datan överfördes via, sedan adderar den ihop all indata och om summan blir över en minimigräns så skickar neuronen talet vidare till nästa neuron[1], detta sker fram tills det sista lagret nås.

Före ett ANN är tränat är vikterna på anslutningarna mellan neuronerna randomiserade[1]. För att lära ett ANN så används väglett lärande eller icke-väglett lärande[1]. Väglett lärande som de flesta ANN har blivit lärda med[1] är när en lärare förser ett data set som är märkt till neuronnätverket, neuronnätet går sedan igenom datasetet och jämför utdatan som den producerade mot den förväntade utdatan, sedan ändras vikterna på anslutningarna och neuronnätet fortsätter köra[1]. Detta fortsätter fram tills neuronnätets producerade utdata är lik de förväntade resultaten[1].

2.2 Agent

En agent är, inom AI, ”allt som kan ses att uppfatta sin miljö genom sensorer och agera på den miljön genom ställdon”[2]. Det finns många olika sorters agenter, en agent som är baserad i mjukvara kan få indata genom tangenttryckningar, och en fysisk agent kan uppfatta sin omgivning med mekaniska sensorer[2]. Komplexiteten på agenter varierar också stort, simpla reflex maskiner som bara har tillgång till miljön nu och komplexa system som agerar baserat på hur miljön är nu och hur den har varit[2].

2.3 Monte Carlo sökträd

Ett Monte Carlo sökträd, på engelska Monte-Carlo Tree Search (MCTS), är en Monte-Carlo baserad bäst först sökning som kan implementeras till alla spel med begränsad längd och fullständig information[3]. Monte Carlo sökträdet infördes först i en GO spelmotor[4] och har sedan dess blivit använd inom flera olika sorters spel för att söka upp optimala spel drag[3].

I ett Monte Carlo sökträd är varje nod ett möjligt speltillstånd och varje löv är ett speltillstånd som inte har blivit simulerat ännu, detta betyder att varje nod är ett unikt lagligt drag i spelet. Roten är det nuvarande speltillståndet och noderna med djupet ett är alla tillstånd som kan nås genom att spelaren gör ett drag. Varje nod som har blivit simulerad innehåller data på hur många drag som resulterar i vinster eller förluster i barnnoderna[3], exempelvis om en nod har tre barnnoder och två av dem är drag som gör att spelaren vinner så har noden data på att två av tre drag resulterar i vinst.

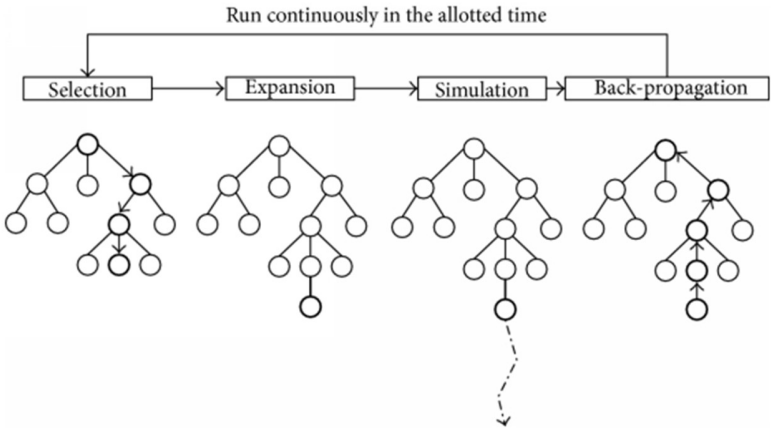
Algoritmen loopar igenom fyra steg för att bestämma ett drag, urval, expansion, simulering, och återkoppling[3]. Algoritmen kör dess fyra steg på flera möjliga speltillstånd, när algoritmen hittar har simulerat alla noder, eller om algoritmen är tidsbunden och den tilldelade tiden tar slut, väljer algoritmen den noden med högst antal simuleringar som det slutgiltiga draget[3].

Steg 1: Urval, Algoritmen startar från trädets rot och går nedåt igenom noderna tills ett löv är nådd.

Steg 2: Expansion, om lövet som är valt inte slutar spelar genom att spelet blir vunnet, förlorat, eller oavgjort så får det valda lövet en eller flera barnnoder.

Steg 3: Simulering, från de nya löven så spelas de simulerade spelen ut fram tills ett resultat fås, dessa drag är slumpmässiga.

Steg 4: Utveckling, använder resultatet från steg 3 för att uppdatera noderna från startnoden i steg 3 till roten.



Figur 2.2. Visuell representation på de fyra steg ett Monte-Carlo sökträd använder.

I spel med få antal drag funkar Monte Carlo sökträd bra men i spel såsom schack blir det extremt tungt för datorn att simulera mångra drag frammåt, man kan lätta på detta genom att begränsa djupet på sökningarna eller att använda policyer för att ignorerar vissa noder som inte är lovande. Eftersom steg 3 använder sig av slumpmässiga drag så är de spelade dragen svaga, för att lindra detta kan man implementera funktioner som gör att sökträder ger mera vikt åt lovande handlingar[3].

2.4 Långt kortvarigt minne

Långt kortvarigt minne (LSTM), på engelska Long short-term memory, är en arkitektur som kan användas i artificiella neuronnät[5]. LSTM används för att delvis lösa problemet med försvinnande och exploderande lutningar, på engelska vanishing/exploding gradient problem, i neuronnät[5].

Problemet med försvinnande/exploderande lutningar är när känsligheten i neuronnätverk, specifikt återkommande neurala nätverk (RNN), förfaller på grund av nya inmatningar [5]. I fallet med försvinnande lutningar så slutar det neurala nätverket lära sig och med exploderande lutningar så tilldelar algoritmen extremt hög betydelse till anslutningar.

3. Maskininlärning

Maskininlärning är ett område som tillhör artificiell intelligens och handlar om att datorn lär sig att göra en eller flera uppgifter som datorn inte är specifikt programmerad att hantera. I detta kapitel förklaras kort vad maskininlärning är och vilka metoder maskininlärning använder.

3.1 Generellt om maskininlärning

Maskininlärning, på engelska machine learning, är en term som myntades av Arthur Samuel år 1959[6]. Maskininlärningsalgoritmer används på många olika områden eftersom det kan underlätta uppgifter som är invecklade eller omöjliga att hårdkoda.

Maskininlärning fungerar genom att bygga upp en modell, som är en representation av en miljö man vill jobba i, genom att läsa tränings-data som den sedan använder för att fatta beslut. En formell definition översatt från engelska på maskininlärning är följande ”*Det sägs att ett datorprogram lär sig av erfarenhet **E** med avseende på en del klasser av uppgifter **T** och prestandamått **P** om dess prestanda vid uppgifter i **T**, mätt med **P**, förbättras med erfarenhet **E**.*”[7].

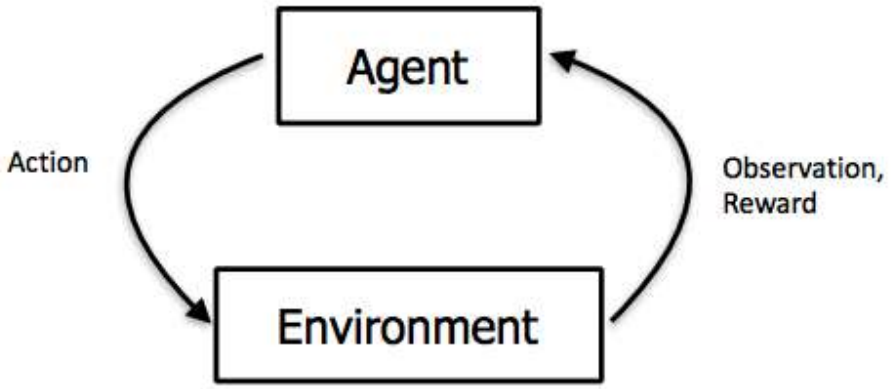
Maskininlärning har flera olika inlärnings algoritmer som är anpassade till olika uppgifter som skiljer sig i vilken sorts indata och utdata algoritmen använder sig av. Dessa metoder har både fördelar och nackdelar som gör att en metod lämpar sig bättre till vissa uppgifter.

Ett problem med maskininlärning är att man inte med säkerhet kan veta varför den gör som den gör, detta problem kallas Svarta lådans problem[8], på engelska black box problem. Med traditionella program vet programmeraren vad programmet gör och varför, med maskininlärning är det mer komplicerat, eller omöjligt, att veta motiveringen bakom hanteringar. Problem som kan uppstå är exempelvis att programmeraren inte kan förbättra programmet när hen inte vet vad som går snett.

3.2 Förstärkningsinlärning

Förstärkningsinlärning är en metod inom maskininlärning där en agent får negativ eller positiv förstärkning baserat på vilka handlingar den utför på en miljö[9]. Målet med detta är att datorn maximera antalet positiva förstärkningar och minimera de negativa[9].

Förstärkningsinlärning funktionerar genom att agenten utför en handling på en miljö, sedan returneras den nya miljön och en belöning som beror på om det var en negativ eller positiv förändring. Baserat på belöningen så får agenten en idé på vilket val som kommer vara optimalt i liknande situationer i framtiden[9].



Figur 3.1. Visuell representation på hur förstärkningsinlärning fungerar.

Eftersom förstärkningsinlärning har många användningsfall och få krav så används det i många olika områden. Med en tillräckligt avancerad förstärkningsinlärnings algoritm är de flesta uppgifterna möjliga.

3.3 Väglett Lärande

Väglett lärande, på engelska supervised learning, är när en maskininlärnings agent använder sig av markerad data som en lärare förser till agenten. Sedan analyserar agenten den markerade datan och hittar mönster som den senare kan använda för att identifiera data som ej är markerad[2].

Ett exempel på väglett lärande är en agent som skall känna igen bilder på bilar, läraren börjar med att mata in ett dataset med bilar som är markerade som bilar. Sedan går agenten igenom bilderna fram tills den når en acceptabel nivå av bildklassificering.

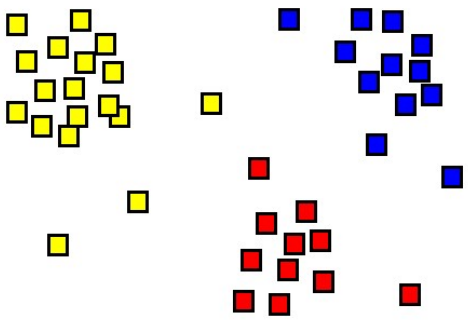
Svarta låde problemet har lindrig verkan på väglett lärande eftersom läraren kan till en viss grad välja vad algoritmen lär sig, dock så lider väglett lärande av flera andra problem. Väglett lärande kan ej klassificera saker den ej har fått exempel på av läraren, och datan som läraren har angivit kan också vara för komplex eller för specifik vilket kan leda till att algoritmen inte känner igen det som den borde kunna klassificera.

Väglett lärande har fördelar och nackdelar som gör denna kategori av maskininlärning specialiserad jämfört med förstärkningsinlärning, det finns också flera olika algoritmer väglett lärande som har varierande prestanda beroende på uppgiften.

3.4 Icke-väglett Lärande

Icke-väglett lärande, på engelska unsupervised learning, är likt väglett lärande förutom att datan som blir försedd inte är markerad, sedan används datan för att hitta okända mönster[10]. Exempelvis så kan algoritmer som använder sig av icke-väglett lärande koppla ihop bilder på hundar av olika raser genom att bara se bilder på en sorts ras.

En vanliga algoritmer som icke-väglett lärande använder sig av är klustring. I klustring används klusteranalysmetoder för att gruppera datapunkter i olika kluster[11], exempelvis så kan bilder på hundar grupperas tillsammans för att de har liknande egenskaper såsom fyra fötter, en svans, och en nos, sedan används dessa kluster för att klassificera bilder beroende på vilket kluster som bilden stämmer in på.



Figur 3.2. En visuell överblick på hur kluster kan vara indelade.

4. Problem med AI i spel

De flesta spelen har någon sorts artificiell intelligens, dock så är de oftast datorprogram med en begränsad förmågan att reagera till olika situationer och utan någon sorts förmåga att lära sig. Detta leder till att artificiell intelligens, specifikt i formen av motståndarspelare och i strategibaserade spel, är svaga och kompenserar genom att fuska eller få orättvisa fördelar.

För att nå toppnivån i spel som är strategi baserade krävs flera egenskaper såsom förmågan att förutse vad motståndarspelaren kommer att göra och intuition. För ett simpelt datorprogram med ett fåtal möjliga reaktioner är det omöjligt att återskapa dessa egenskaper på ett sådant sätt som skulle möjliggöra att programmet kan funktionera som en kapabel motståndare i ett komplext spel.

Användningen av maskininlärning kan göra så att artificiell intelligens i spel kan bli tillräckligt kapabla att spela mot mänskliga spelare på toppnivå. Optimalt skulle artificiell intelligens i spel vara lika begränsade som mänskliga spelare så att den vinner för att den spelar smartare och bättre, i flera spel skulle det vara trivialt för en AI att vinna om den har extremt snabb reaktionsförmåga och perfekt information vad som händer i spelet.

Det finns också flera komplikationer som gör det svårare för AI att funktionera beroende på vilken sorts spel den spelar, exempelvis schack är ett spel där bara en spelare gör ett drag åt gången och som har fullständig information, vilket betyder att båda spelarna vet vart alla pjäser som är med i spelet befinner sig för tillfället. Detta är optimalt för en dator eftersom den har tid att köra beräkningar och den vet vart pjäserna befinner sig så det är trivialt att använda dem i beräkningen, dock om ett spel har ofullständig information och spelas i realtid så måste datorn utföra alla beräkningar i realtid och förutspå vart fiendespelaren har sina spelpjäser.

5. AI med maskininlärning i strategibaserade spel

Väldigt få spel har AI som använder sig av maskininlärning för att förbättra sig, dock finns det flera företag som forskar i AI och maskininlärning som utvecklar AI till spel för att göra framsteg i sin forskning. I detta kapitel undersöks artificiell intelligens som har blivit utvecklade för att spela strategibaserade spel.

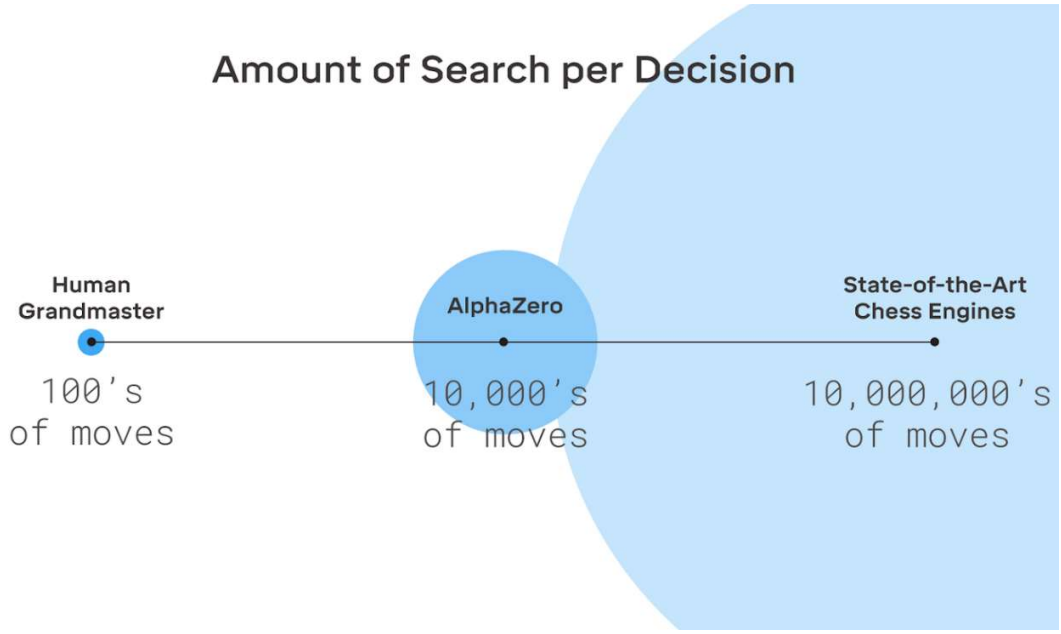
5.1 AlphaZero

AlphaZero är en AI som spelar schack, Shogi och Go som blev utvecklad av DeepMind i slutet av 2017[12]. DeepMind hade tidigare utvecklat AI till brädspelen dock så var AlphaZero deras första AI som spelade alla tre spel. AlphaZero's algoritm är en generaliserad version av den som AlphaGo Zero, DeepMinds AI som spelade GO, använde.

AlphaZero lär sig spela med användning av ett otränat artificiellt neuronnät, vilket bara har tillgång till reglerna, som med förstärkningsinlärning spelar miljontals spel mot sig själv[12]. I början gjorde AlphaZero bara slumpmässiga drag men ju mera den spelade

desto vettigare drag börjar den använda sig av[12]. Mängden tid för AlphaZero att lära sig spela ett spel optimalt varierar baserat på spelet ungefär nio timmar för schack, tolv timmar för Shogi, och 13 dagar för Go[12].

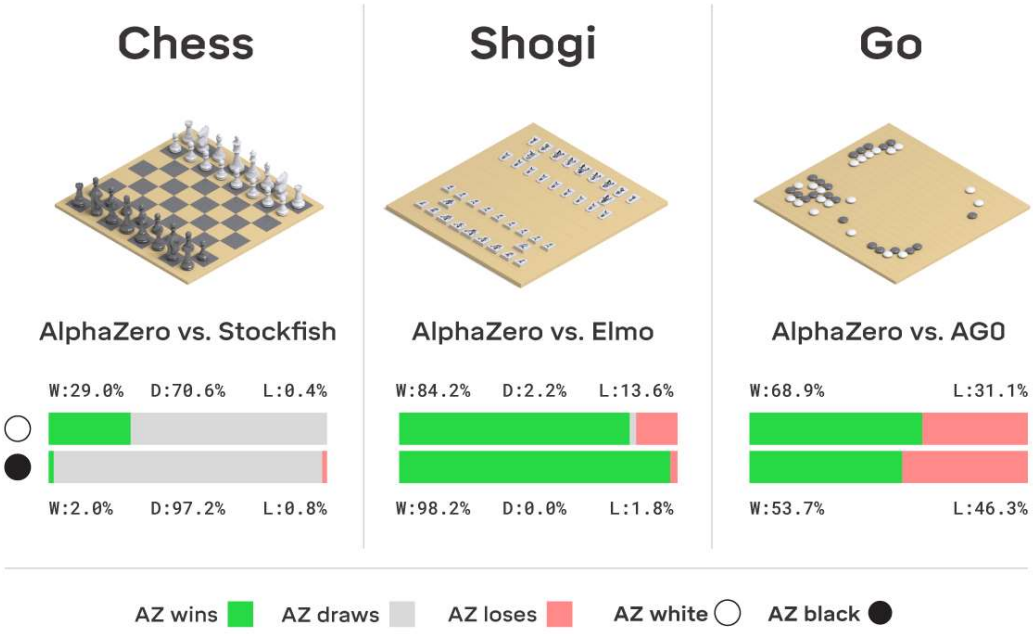
Efter att det artificiella neuronnätet har blivit tränat så använder AlphaZero sig av ett Monte-Carlo sökträd för att söka fram det mest fördelaktiga draget, det artificiella neuronnätet används för att avgränsa sökningen i Monte-Carlo trädet till de bästa dragen, detta gör att AlphaZero söker igenom relativt lite drag i jämförelse med spelmotorer i schack, Go och Shogi[12]. I figur 5.1 kan man se att StockFish, som är en schackmotor utan maskininlärning, söker ungefär 1000 gånger flera drag per sekund i jämförelse med AlphaZero[12].



Figur 5.1. Visuell representation på antalet drag AlphaZero söker igenom jämfört med StockFish och en mänsklig mästare.

Efter AlphaZero ansågs vara färdig så spelade den mot de främsta datorspelarna i schack, Shogi och Go. Före dessa matcher tränades AlphaZero med hjälp av 5,000 tensorbehandlingsenheter (TPU) men under matcherna kördes den på fyra TPU:s och en 44-kärnig processor[12].

I schack spelade AlphaZero mot StockFish 8 som var världsmästaren i Top Chess Engine Championship (TCEC) 2016[12]. AlphaZero tränades i nio timmar före spelen. AlphaZero spelade 1000 matcher mot StockFish 8, i figur 5.2 nedan ser man att AlphaZero som vit vann 29% och förlorade 0,4% av spelen, som svart vann AlphaZero 2% och förlorade 0,8% av spelen, resten av spelen blev oavgjorda[12].

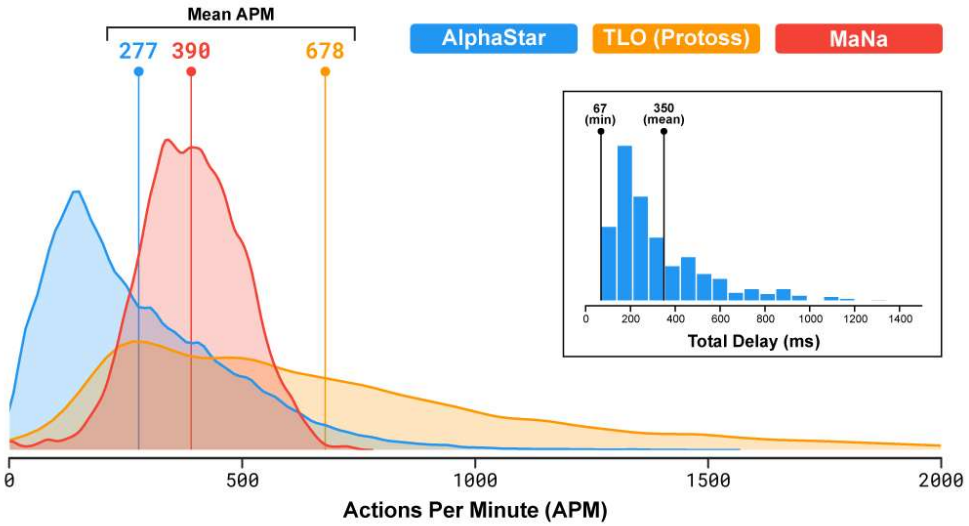


Figur 5.2. Vinstfrekvens mot StockFish, Elmo, och AlphaGo Zero.

5.2 AlphaStar

AlphaStar är en AI lagad av DeepMind som spelar realtidsstrategi spelet StarCraft 2[13]. Efter att DeepMind lagade AlphaZero ansågs nästa steg vara StarCraft eftersom spelet har flera svårigheter som AlphaStar måste övervinna såsom ofullständig information och turer i realtid[13].

För att undvika att AlphaStar vinner på grund av en dators naturliga fördelar så begränsade DeepMind AlphaStars reaktionsförmåga och ”Actions-Per-Minute”(APM), som står för hur många handlingar AlphaStar kan utfärda till spelet per minute, till en mänsklig nivå[13]. I figuren nedan kan man observera att AlphaStars genomsnittliga APM var märkbart lägre än TLO och MaNa som är professionella StarCraft spelare. AlphaStars reaktionstid var genomsnittligen 350ms[13]. Den första versionen av AlphaStar använde sig av rådata som den fick från StarCraft, detta gjorde att den heltiden såg alla sina enheter, eftersom en mänsklig spelare är beroende på att flytta en kamera för att se sina enheter så begränsades AlphaStars synfält i en senare version med tilläggningen av en artificiell kamera som AlphaStar var beroende av[13].

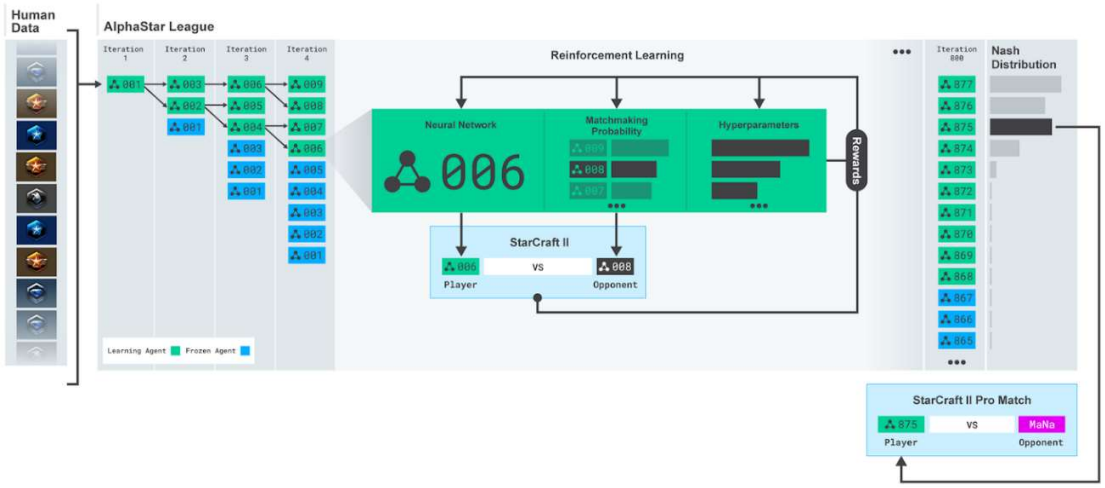


Figur 5.3. AlphaStars APM och reaktionstid.

För att bestämma vilka handlingar den gör använder sig AlphaStar sig av ett djupt artificiellt neuronnät som i indata får en lista med enheter och deras egenskaper, sedan använder AlphaStar det neurala nätverket för att bestämma vilken handling som är bäst[13]. En mera specifik definition av det artificiella neuronnätet av DeepMind lyder: ”*The neural network architecture applies a transformer torso to the units (similar to relational deep reinforcement learning), combined with a deep LSTM core, an auto-regressive policy head with a pointer network, and a centralised value baseline.”* [13]

Initialt lärde sig AlphaStar med väglett lärande genom att DeepMind matade in StarCraft spel som spelats med människa mot människa[13], efter att AlphaStar hade lärt sig att imitera strategierna av en mänsklig spelare och kunde spela spelet på en viss nivå började AlphaStar lära sig med förstärkningsinlärning[13]. AlphaStar används sig av ”AlphaStar League” som är en liga där flera agenter av AlphaStar spelade mot varandra.

I figur 5.4 nedan ser man hur ligan grenar ut de gamla agenterna och sedan fryser dem så att agenterna hela tiden blir bättre med hjälp av förstärkningsinlärning när de spelar mot andra agenter. För att agenterna skall fortsätta förbättra sig utan att glömma bort gamla strategier blev de individuella agenternas parametrar ändrade så att agenterna hade olika mål, detta påverkade hur de spelade och vilka strategier de använde sig av, vilket gjorde att varje iteration hade diversifierade matcher och inte agenter som bara använde sig av en begränsad samling strategier[13].



Figur 5.4. Visuell representation på AlphaStar League.

För att testa AlphaStars spelförmåga anordnade DeepMind tio matcher mot två professionella StarCraft spelare, AlphaStar vann alla tio matcher[13]. Spelarna kommenterade att AlphaStar hade en mänsklig spelstil men använde sig av unika strategier som inte hade blivit använda förr[13].

5.3 OpenAI Five

OpenAI Five är ett projekt som består av fem artificiella neuronnät som spelar lagspelet Dota 2[14]. OpenAI som är utvecklarna av OpenAI Five, har tidigare lagat en AI som spelat Dota 2, dock var den bara kapabel att spela ensam mot en singulär motståndare och med speciella regler[14]. För att spela i ett lag och med de normala tävlingsinriktade reglerna, med ett få undantag, så behövde OpenAI Five lära sig använda samarbete och strategi.

OpenAI Five använder sig av förstärkningsinlärning med direkt policysökning, OpenAI Five körs på 256 grafikprocessorer och 128,000 processor kärnor[14]. OpenAI Five spelar upp till ~180 år, ~900 år med alla 5 neuronnätverk medräknat, av Dota 2 spel mot sig själv dagligen[14].

Före man börjar ett Dota 2 spel så väljer varje spelare en egen spelkaraktär, också kallad hjälte. Varje hjälte har unika förmågor som innebär att spelaren måste spela på olika sätt beroende på vilken hjälte de har, för att hantera detta så använder OpenAI Five skilda Långt kortvarigt minnes(LSTM) nätverk för varje hjälte[14].

Eftersom OpenAI Five använder förstärkningsinlärning från grunden så började agenterna med att mållöst vandra omkring på kartan, efter några timmars spelande så hade agenterna börjat använda sig av grundläggande tekniker och efter några dagar så använde sig agenterna av strategier som var jämförbara med strategier som mänskliga spelare använder sig av[14].

Så att inte agenterna glömmer bort gamla strategier så spelar OpenAI Five 80% av spelen mot en aktuell version av sig själv och 20% av spelen mot äldre versioner, detta gör så att de OpenAI Five blir påmind om strategierna som de äldre versionerna använder[14].

OpenAI Five har några fördelar som en mänsklig spelare inte har som kan ha en inverkan på hur bra de spelar, agenterna gör inte i misstag handlingar, som människor ofta gör, och deras reaktionstid är i genomsnitt 80ms[14] som är märkbart lägre än en mänsklig spelares.

Vid OpenAI Five Finals, som var en turnering OpenAI anordnade för att visa kapabiliteten av OpenAI Five, spelades en uppvisningsmatch där OpenAI Five spelade en bäst av tre mot OG[15], ett att de bästa Dota 2 lagen som har vunnit The International, vilket är det största turneringen inom Dota 2, år 2018 och 2019. OpenAI Five vann matchen med relativ lätthet[15]. OpenAI Five utvecklas inte mera av OpenAI dock så använder OpenAI teknologin som de tog i bruk i sina fortsatta projekt[14].

6. Avslutning

Det bästa alternativet för utvecklingen av AI i strategi baserade spel som vinner för att de spelar strategiskt och effektivt är med implementeringen av någon metod av maskininlärning. Förstärkningsinlärning är ett effektivt sätt att träna AI och kommer användas i artificiell intelligens framöver.

Bara ett fåtal spel har någon sorts AI som använder maskininlärning eftersom det ännu inte skulle vara så effektivt att implementera maskininlärning i varje AI på grund av utvecklingstiden och datorprestandan som krävs för att utveckla en AI till ett specifikt spel. Dock så finns det några företag som går i spetsen för utvecklingen av AI med maskininlärning och småningom kommer någon skapa en AI som kan implementeras till flera spel och som kräver relativt lite datorprestanda för att få inträna.

De AI som använder maskininlärning och existerar har redan visat att det är möjligt att spela på toppnivå i även de mest komplexa spelen, och i flera spel är en AI redan bättre än alla mänskliga spelare.

Det finns en stor chans att användningen av AI som har maskininlärning kan bli en norm inom den närmaste framtiden. De framsteg som uppkommer vid utvecklingen av AI kan också ha stor inverkning på AI och maskininlärnings forskningen överlag.

7. Figurer

Figur 2.1: Glosser.ca, *Colored neural network*, 2013. Wikipedia Commons
Hämtad 31.3.2020 från:
https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Figur 2.2: Świechowsk, M., HyunSoo, P., Mańdziuk, J., Kyung-Joong, K., *Recent Advances in General Game Playing,* 2015. Hindawi.
Hämtad 31.3.2020 från:
https://www.hindawi.com/journals/tswj/2015/986262/

Figur 3.1: Omanakuttan Prasanth, *What is inverse reinforcement learning?*, 2018. GOODAUDIENCE
Hämtad 31.3.2020 från:
https://blog.goodaudience.com/what-is-inverse-reinforcement-learning-e333228af146

Figur 3.2: hellisp, *Cluster-2*, 2006. Wikipedia Commons
Hämtad 31.3.2020 från:
https://commons.wikimedia.org/wiki/File:Cluster-2.png

Figur 5.1, 5.2: Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D., *AlphaZero: Shedding new light on chess, shogi, and Go,* 2018. Deepmind
Hämtad 31.3.2020 från:
https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go

Figur 5.3, 5.4: Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., & Silver, D.. ”*AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*”, 2019.
Hämtad 31.3.2020 från:
https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

8. Källor

[1] Anderson, D., McNeill, G., *Artificial neural networks technology*, 1992. Hämtad 31.3.2020 från: http://andrei.clubcisco.ro/cursuri/f/f-sym/5master/aac-nnga/AI_neural_nets.pdf

[2] Russell Stuart, Norvig Peter, *Artificial Intelligence: A Modern Approach, Third edition,* 2009.

[3] Chaslot, G., Bakkes, S., Szita, I., Spronck, P., *Monte-Carlo Tree Search: A New Framework for Game AI,* 2008.

[4] Silver, D., Huang, A., Maddison, C. et al. *Mastering the game of Go with deep neural networks and tree search*. Nature 529, 484–489 (2016). Hämtad 31.3.2020 från: https://doi.org/10.1038/nature16961

[5] Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-term Memory*, 1997. Neural computation, 9, 1735-80.

[6] Arthur L. Samuel (1959). *Some studies in machine learning using the game of Checkers,* 1959. IBM JOURNAL OF RESEARCH AND DEVELOPMENT, 71–105.

[7] Mitchell Tom, *Machine Learning,* 1997. McGraw Hill.

[8] Carlos Zednik (2019). *Solving the Black Box Problem: A General-Purpose Recipe for Explainable Artificial Intelligence,* 2019. CoRR, abs/1903.04361.

[9] Sutton, R., & Barto, A, *Reinforcement Learning: An Introduction*, 2018. The MIT Press.

[10] Hinton G., S. *Unsupervised Learning: Foundations of Neural Computation*, 1999. MIT.

[11] Sazonov, E. (2010). Clustering (Xu, R. and Wunsch, D.C.; 2008) [Book review]IEEE Pulse, 1, 74-76.

[12] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D., *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*, 2018. Science, 362(6419), 1140–1144.

[13] Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., & Silver, D., *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*, 2019. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

[14] OpenAI, *OpenAI Five*, 2018.
https://openai.com/blog/openai-five/

[15] OpenAI, *OpenAI Five Finals*, 2019.
https://www.twitch.tv/videos/410533063?t=44m53s