Maskininlärning för bottar i realtidsstrategispel

Jacob Rosing

Kandidatavhandling i Datavetenskap

Åbo Akademi – Fakulteten för naturvetenskaper

Handledare: Luigia Petre

[Datum]

Referat

Maskininlärning är ett delområde av artificiell intelligens (AI) som går ut på att, med hjälp av olika inlärningsmodeller, få datasystem att lära sig lösa problem på egen hand samt lära sig av egna upplevelser. Ett område där maskininlärning kan tillämpas utan att behöva ta i beaktande etiska aspekter och säkerhetsaspekter är datorspel. Bland dessa är det kanske främst strategispel som kan dra särskild nytta av kraftfullare AI-algoritmer eftersom dessa spel lägger större tyngd på strategiskt tänkande än mekanisk skicklighet som redan i nuläget kan fulländas utan maskininlärning. Avhandlingen går igenom maskininlärning i realtidsstrategispel över lag med fokus på StarCraft II och den artificiella intelligensen AlphaStar som spelar det förstnämnda. Tillämpningar av lärdomar från artificiella intelligenser á la AlphaStar inom övriga områden såsom autonoma fordon berörs även kortfattat.

Sökord: artificiell intelligens, spel, bottar, maskininlärning, djupinlärning, realtidsstrategispel

Innehållsförteckning

1. Inledning

Anledningen till ämnesvalet är delvis bristen på helt svenskspråkiga avhandlingar som redogör för maskininlärning inom detta specifika tillämpningsområde. En annan anledning till att detta ämne valdes, som kanske även är den främsta anledningen, är mitt eget intresse för både RTS-spel och artificiell intelligens.

1.1 Avgränsning

Maskininlärning är ett delområde av artificiell intelligens som går ut på att få datorsystem att lösa problem på egen hand och lära sig av egna upplevelser.[1]

Realtidsstrategispel eller realtidsstrategi (RTS) är en datorspelsgenre som skiljer sig från övriga strategispel genom att samtliga drag görs i realtid och inte i vändor.[3] I och med att strategier i realtidsspel kan bli väldigt komplexa så utgör spel ur genren ett bra område att utveckla kraftfulla AI-algoritmer för.[2]

Ibland används, inom realtidsstrategispel, underlättande ändringar av spelreglerna för att kompensera för bottars bristande problemlösningsförmåga, men denna avhandling kommer i stället att handla om användning av maskininlärning för att skapa utmanande bottar i denna typ av spel. Genom att bottar använder sig av maskininlärning så kan de alltså besegra skickliga motståndare genom att spela smartare än traditionellt kodade bottar kan. Stor tyngd kommer att läggas på spelet StarCraft II eftersom det finns en artificiell intelligens som kan spela det på hög nivå (AlphaStar) [2], samt eftersom spelets inbyggda AI utgör ett bra exempel på en bott som kompenserar för sin avsaknad av skicklighet med extra resurser samt total synlighet av spelplanen.

1.2 Syfte

Avhandlingen kommer att ta upp olika maskininlärningsprojekt inom RTS spel, samt teknologin som dessa projekt använder sig av. Projekten kommer att jämföras med varandra för att få en inblick i deras metoder och algoritmer för att se vad de har gemensamt och var de skiljer sig åt samt varför de gör det. Avslutningsvis så kommer möjligheten att tillämpa lärdomar från dessa projekt inom icke-spelrelaterade områden att diskuteras.

2. Artificiella neuronnät

För att förstå sig på de maskininlärda intelligenser som byggts i RTS-spel så krävs en viss förståelse för algoritmerna som använts. De flitigast använda tillvägagångssätten faller under samlingsnamnet artificiella neuronnät (ANN) som omfattar de arkitekturer vars uppbyggnad, med hjälp av s.k. noder, efterliknar en hjärnas nätverk av neuroner. Noderna tar emot signaler i form av reella tal, bearbetar dem och skickar sedan vidare dem till följande nod. Enskilda nod-förbindelsers påverkan justeras när nätverket tränas.[4] En nackdel med neuronnät är att det kan vara svårt att utföra debugging, d.v.s. felsökning, på dem på samma sätt som man debuggar traditionell kod eftersom de är som svarta lådor i och med att det är svårt att se hur de transformerar indata till utdata. Om ett neuronnät inte fungerar som tänkt så behöver man i stället granska indata, utdata och hur ens nät tränas.[5]

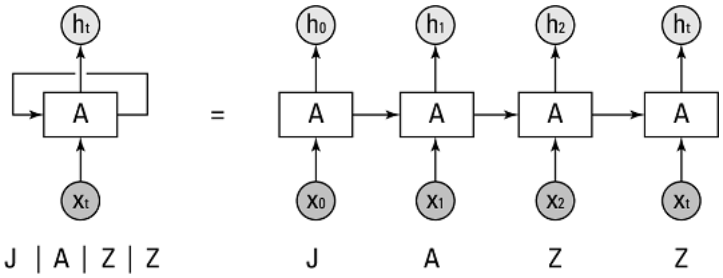
2.1 Djupinlärning

Djupinlärning skiljer sig från traditionell maskininlärning genom att ansvaret för extraktion av egenskaper förs över från människa till maskin. Delen ”djup” i djupinlärning syftar på användandet av flera lager (oftast neuronnät) som omvandlar sin indata till ett mer abstrakt format. Ett exempel på detta är när en bildigenkänningsmodell som använder sig av djupinlärning ska känna igen ett ansikte i en bild. Modellen börjar i exemplet med att abstrahera pixlar och inkoda kanter för att i påföljande lager inkoda kantgrupper. Påföljande lager identifierar ansiktskomponenter såsom ögon och näsor vilket sedan används för att identifiera ansikten i bilden. Något som inte kan skötas av modellen själv [källa behövs] är justering av antal lager samt deras storlek, vilka är saker som påverkar till vilken grad modellen kan abstrahera mönster i bilder.[6]

2.2 RNN

Ett återkopplande neuralt nätverk (RNN), eller recurrent neural network som det heter på engelska, är en typ av neuronnät där behandlad information består och därmed kan användas när nätverket hanterar efterföljande informationsstycken. Nätet får alltså två indata: ny information som ska behandlas, och den information som behandlats direkt innan. Eftersom nätverket hanterar datasekvenser i ordning och har ett minne så är det bra

för bland annat prediktiva tillämpningar men även situationer där ordning av data spelar roll, såsom exempelvis taligenkänning, där ordens ordning påverkar betydelsen av meningar. En nackdel med traditionella RNN nät är att de har en oförmåga att koppla ihop två relaterade stycken information som befinner sig långt ifrån varandra i informationsflödet. Näten glömmer alltså gradvis tidigare inmatade sekvensdelar när de kalkylerar utdata av nyare delar. Detta orsakas av det s.k. försvinnande gradientproblemet (vanishing gradient problem på engelska) som orsakas av gradientbaserade inlärningsmetoder där var träningssession medför en justerad påverkan av nodförbindelser. Ifall gradienten blir för liten, som den ofta blir när man använder sig av många lager, så kan justeringen av nodförbindelsernas påverkan bli så liten att neuronnätets träning i praktik upphör. På grund av dess bristande långtidsminne så används sällan icke-modifierade versioner av RNN, utan nyare implementationer såsom LSTM föredras oftast. [1] [8]



Bilden ovan illustrerar ett RNN näts minnesloop när bokstäverna som tillsammans bildar ordet JAZZ fås som indata [13]

2.2.1 LSTM

LSTM (från engelskans Long short-term memory) är en sorts vidareutveckling av RNN nät som presterar bättre än vanliga RNN nät när information behöver kommas ihåg över längre tidsramar. Det försvinnande gradientproblemet som vanliga RNN nät lider av löses alltså eftersom gradienten/lutningen hålls brant/hög nog för att träningen ska hållas kort och precisionen hög.

Ett LSTM nät lagrar information i en sorts minnescell som avgör huruvida information bör lagras eller ej beroende på hur viktig informationen anses vara. Informationens betydelse avgörs av en vikt som tilldelas av algoritmen.

LSTM nätets minnescell har tre portar som indata till nätet går via. En indata-, en utdata- och en glöm-port. Indataporten sparar informationen, utdataporten genererar utdata från cellens innehåll och glömporten raderar onödig information.[8]

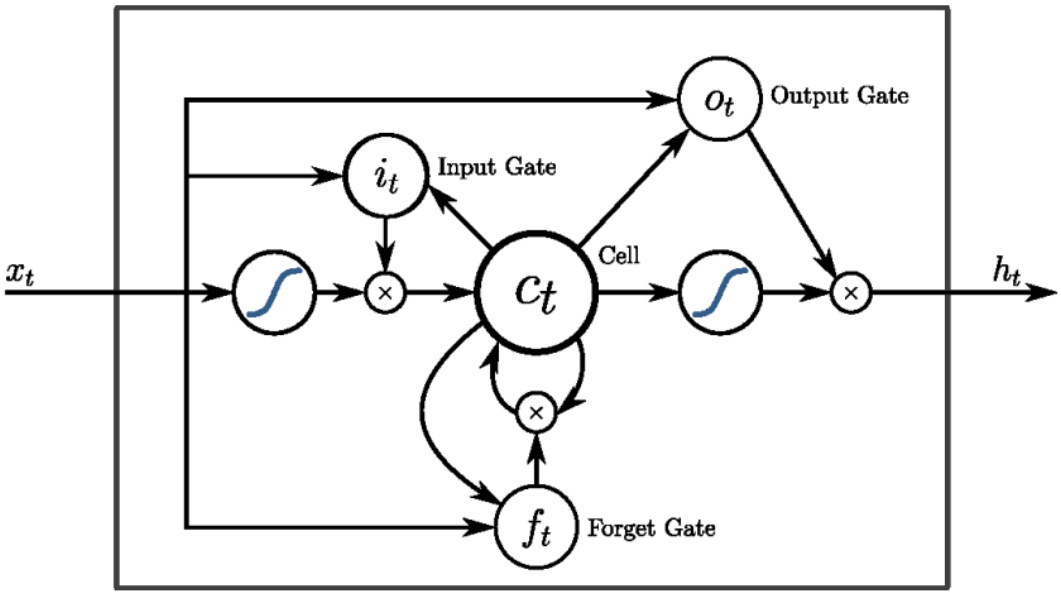


Illustration av LSTM nätets tre portar [14]

2.3 MLP

En MLP är ett sorts artificiellt neuronnät som består av minst tre lager noder (indata-, gömt- och utdata-lager) och som använder sig av en övervakad inlärningsteknik med förmåga att ge ”prognoser om framtida händelser med hjälp av ingångsdata”.

När en MLP lär sig så styrs storleken på nätets uppdateringar av en sorts inlärningstakts-parameter. Med en högre inlärningstakt så går träningen snabbare, dock med en ökad risk för problem såsom en extremt liten eller extremt stor gradient. En låg inlärningstakt kan däremot medföra att inlärningen tar allt för länge.

Utöver inlärningstakt så finns en parameter som kallas momentum. Momentum ökar inlärningstakten genom att accelerera förändringar som skett flera gånger i rad. Denna parameter bör man däremot vara försiktig med i senare skeden av träningssessioner eftersom momentum kan orsaka liknande problem som en hög inlärningstakt kan. [7]

3. Förstärkningsinlärning

Förstärkningsinlärning är en maskininlärningsmetod som går ut på att belöna önskvärt beteende och/eller straffa oönskat beteende. Agenten, dvs AIn under träning, som tränas med förstärkningsinlärning försöker alltså maximera mängden belöningar den får. Detta sker genom att en s.k. policy avgör vad för beslut agenten ska ta under specifika omständigheter. Policyn kan vara deterministisk eller stokastisk. Om en policy är deterministisk så finns det en specifik handling för varje tillstånd. En stokastisk policy tilldelar emellertid icke-noll sannolikhetsvärden med summan 1 för varje handling i varje tillstånd. Vilken handling som väljs beror endast på det aktuella tillståndet.

Agenten kan välja att antingen utforska eller utnyttja. Utforskning innebär att agenten lär sig mer om sin omgivning för att på så sätt bättre kunna väga för och nackdelar med de beslut den kan ta. Utnyttjning innebär å andra sidan att agenten tar de beslut som den tror att kommer leda till högsta möjliga belöningsmängd på lång sikt. Fastän inlärningsmetoden inte kräver märkt data så kan den använda sig av det för att minska på mängden utforskning som krävs.

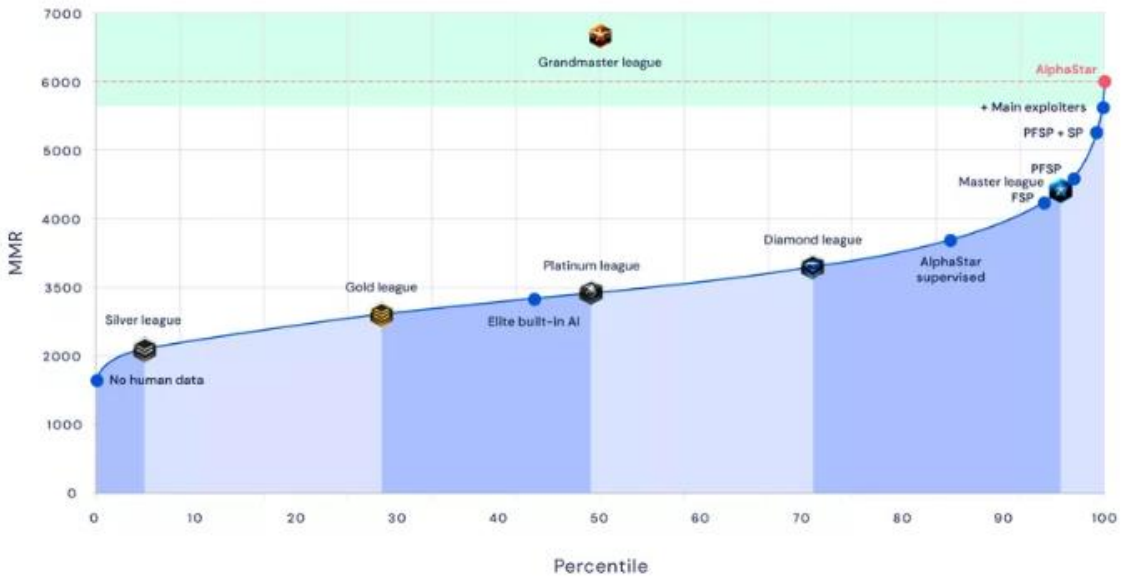
För lösa en uppgift med förstärkningsinlärning så krävs det alltså att man hittar en policy som ger en stor mängd kumulativa belöningar. Bland maskininlärda bottar är förstärkningsinlärning den träningsmetod som används flitigast. [9]

3.1 Proximal policyoptimering

Proximal policyoptimering (PPO) är en sorts övervakad förstärkningsinlärningsalgoritm som lägger mjuka begränsningar på hur stora skillnader en ny policy kan ha jämfört med den som kom innan, vilket bland annat avsevärt minskar risken för stora försämringar av policyn i de fall där en stor förändring har en icke önskvärd effekt. Detta är fördelaktigt när man vill hålla träningens varians låg samt när en minimal förlustfunktion, vilket tyder på en optimerad modell, önskas. [10]

4. AlphaStar

AlphaStar, ett datorprogram som använder sig av djupinlärning för att spela StarCraft II, är troligtvis det främsta exemplet på en lyckad tillämpning av maskininlärning för en bott i ett realtidsstrategispel, om inte rentav i ett spel över lag. AlphaStar utvecklades av DeepMind Technologies, ett brittiskt företag som specialiserar sig på utveckling av artificiella intelligenser och den första testmatchen mot en professionell StarCraft spelare hölls 2018. Testmatchen vanns av AlphaStar som dock hade ett övertag i och med att den kunde se hela spelkartan, dock med s.k. krigsdimma, på en gång samt hade omänskligt bra precision och kvickhet när den klickade i spelet. Ungefär ett år senare hade botten nått den högsta skicklighetsnivån i spelet. Under tiden som AlphaStar spelade matcher för att nå ”grandmaster” nivå, som den högsta skicklighetsnivån kallas, så hade den dock fått begränsningar i hur snabbt den kunde klicka och dessutom så använde den sig nu av samma kameravy som mänskliga spelare för att få indata, vilket gav den mer mänskliga förutsättningar. [2]



Bilden ovan visar olika iterationer av AlphaStar i förhållande till spelets inbyggda bottar samt de olika skicklighetsrangerna [2]

4.1 StarCraft II

StarCraft II är ett realtidsstrategispel med science fiction-tema som släpptes 2010. Spelets flerspelarläge var tidigare föremål för världens största e-sportsscen och har, vid avhandlingens skrivande, ännu en aktiv spelarbas.

Spelet lägger stor tyngd på strategiskt och taktiskt tänkande samt mekanisk skicklighet, vilket främst mäts i handlingar per minut (APM, från engelskans ”actions per minute”). Spelaren (eller spelarna) väljer att spela som en av tre raser (Terran, Zerg och Protoss) med olika styrkor och svagheter. Det huvudsakliga målet är att förstöra motståndarens enheter och byggnader. Spelaren börjar med ett fåtal arbetar-enheter vars huvudsyfte är att samla resurser och bygga byggnader, som i sin tur producerar och uppgraderar enheter. Den ekonomiska aspekten av spelet kallas för makro (”macro-management”) och det individuella styrandet av enheter, eller grupper av enheter, kallas mikro (”micro-management”). Spelarens information om sin omgivnings tillstånd är ofullständig, d.v.s. man kan se terrängen och spelkartans resurser men kartan täcks av en s.k. krigsdimma. Krigsdimman innebär att motståndarens samtliga enheter och byggnader (och därmed

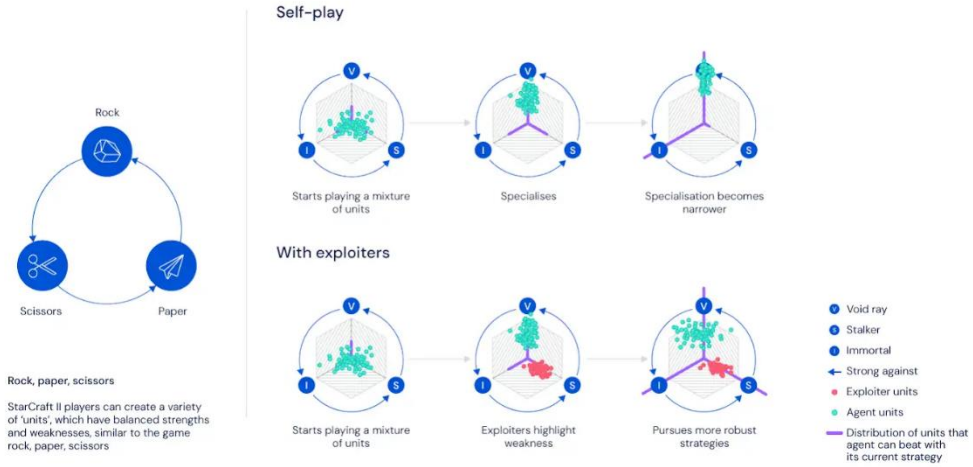
även dess planer) förblir dolda fram tills spelaren med hjälp av sina egna enheter samlat information om dem. [11]

4.2 Algoritmen

AlphaStars beslut tas av ett djupt neuronnät som initialt tränades med hjälp av övervakad inlärning som användes med mänskligt spelade matcher.

Botten imiterade alltså drag gjorda av mänskliga spelare för att lära sig nya grundläggande strategier. Detta tillvägagångssätt är viktigt eftersom processen att hitta nya drag på egen hand och sedan se vilka som fungerar, enligt skaparna, skulle ta alldeles för länge. Dessutom säkerställde detta att en mångfald strategier och taktiker testades. Efter imitationsfasen så övade de s.k. agenterna som tränats på detta sätt mot olika varianter av varandra samtidigt som förstärkningsinlärning tillämpades med vunna matcher som belöning. Agenterna spelade även mot ”exploiter agents” (ung. utnyttjaragenter) som försökte finna och utnyttja svagheter i de huvudsakliga agenternas sätt att spela. På detta sätt upptäcktes optimala strategier och motstrategier samtidigt som eventuella svagheter

åtgärdades. Den slutliga AlphaStar agenten bestod i sin tur av den mest effektiva kombinationen av strategier som upptäckts.

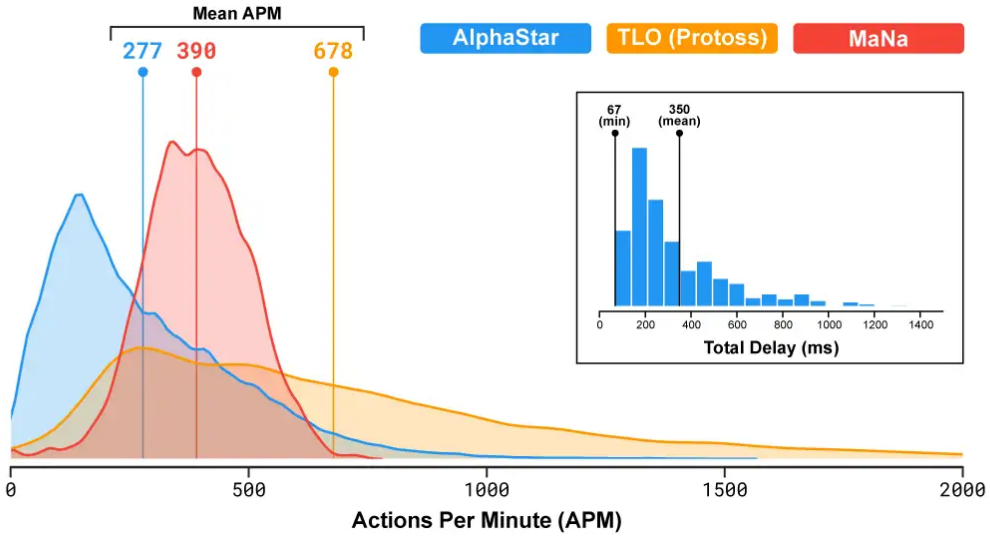


AlphaStar-agenters enhetsfördelnings förändring efter att ha spelat mot sig själva samt ”exploiter”-agenter [2]

Indata till neuronnätet kommer i form av observationer som görs från spelkartan. Utdata i sin tur kommer i form av de handlingar som neuronnätet valt baserat på informationen

den fått som indata. Var och ett av AlphaStars beslutssteg har ungefär 10^{26} möjliga handlingar.

För att botten inte skulle få några orättvisa fördelar på en mekanisk nivå så begränsades dess APM. Dessutom så lider botten av nätverkslatens och beräkningstider som eliminerar en del av datorers inneboende övertag mot människor. [2]



AlphaStar har, som bilden ovan visar, lägre APM än professionella spelare som den besegrat [2]

4.3 Lärdomar

Enligt skaparna av AlphaStar så kan framsteg som gjorts inom artificiella intelligenser som deras eget komma till nytta i utveckling av autonoma fordon och robotar, eftersom de ofta kräver beslut i realtid i miljöer där man inte kan få fullständig information om allt som sker i omgivningen. De nämner också att dessa artificiella intelligenser kan stöta på oväntade händelser som de kan behöva reagera på. Dessutom anser de att förstärkningsinlärningsalgoritmen som använts till AlphaStar kan användas i varje tänkbar inlärningsmiljö. Utöver det så nämner de också bland annat användning av utnyttjaragenter som en teknik som kan användas till alla sorters maskininlärningsproblem inom flerspelarområden. [2]

5. OpenAI Five

OpenAI Five är ytterligare ett exempel på ett lyckat maskininlärningsprojekt i spelvärlden. Detta projekt utgörs av fem bottar som spelar Dota 2, som är ett spel i MOBA (även kallat Action-RTS) genren. Projektet presenterades för första gången åt allmänheten år 2017 då det bestod av en bott som spelade Dota 2 i en mot en matcher. I april 2019 lyckades de, vid det laget, fem bottarna besegra spelets bästa lag. Företaget OpenAI hade alltså inte endast lyckats skapa en bott som presterade väl individuellt. De hade även konstruerat bottar som var ytterst kapabla till samarbete. [12]

5.1 Dota 2

Dota 2 är som tidigare nämnt ett s.k. MOBA, alternativt Action-RTS spel. Fastän det inte är ett renodlat realtidsstrategispel så har det en hel del likheter med exempelvis StarCraft. Dessa likheter är bland annat mikrohantering av enheter, spelarens vy av spelkartan samt avsaknad av konstant fullständig information eftersom spelkartan är täckt av krigsdimma. Skillnaderna mellan spelen är bland annat att Dota 2 är ett lagspel med lag bestående av fem spelare (StarCraft kan också var det men oftast inte) … Spelarna kan välja bland ett flertal karaktärer, med olika förmågor, att spela som. Utöver spelarna så har var lag även statiska torn (kallade towers) som anfaller fiender i sin närhet och mindre datorstyrda enheter (kallade creeps) som går mot fiendens bas och anfaller alla fiendeenheter och byggnader i sin väg. Det finns även en del andra enheter i spelet som dock inte är av större intresse i denna avhandling. Var lag har ett par towers och flera creeps som skapas kontinuerligt. Valutan som används i spelet för att köpa föremål eller omedelbart återuppliva sin karaktär (något som annars tar ett tag) kallas gold. Förutom att köpa bättre föremål åt sin karaktär med guld, så kan man även förbättra sin karaktär genom att öka dess ”level”, vilket kan beskrivas som karaktärens styrkenivå. Detta görs genom att samla ”experience” genom att få in dödande slag (i Dota kallat last hit) på fiendeenheter eller vara nära när någon annan gör det.

5.2 Algoritmen

OpenAI Five utgörs av fem neuronnät som under flera månaders tid spelade 180 år värt av matcher mot sig själv per dag. Till skillnad från AlphaStar så lär sig OpenAI Five sig helt själv, d.v.s. ingen imitation av mänskliga spelares drag görs, och i början är neuronnätens vikter slumpmässigt valda. Förstärkningsinlärningsalgoritmen PPO körs på flera hundra grafikprocessorer samt över hundra tusen processorkärnor. Med separata LSTM nätverk för varje spelkaraktär så lär sig de neurala nätverken olika strategier. Agenten spelar 80% av sina matcher mot sig själv och 20% mot tidigare versioner av sig själv. Detta tillvägagångssätt används för att undvika strategikollaps, d.v.s. att agenten glömmer hur man spelar mot flera olika motståndare eftersom endast en snäv mängd strategier krävs för att besegra sin föregångare. [12]

Agenternas evolution såg ut som så att de under sina första par matcher bara vandrade mållöst runt spelkartan. Under tiden de tränades så började de dock lära sig enklare spelkoncept och efter ett tag så hade de lärt sig spelet så pass bra att de kunde utföra strategier på hög nivå. [12]

Själva samarbetet sker inte via någon sorts kommunikationskanal mellan näten, utan det styrs av en hyperparameter mellan noll och ett som styr hur stor vikt agenterna bör lägga på sin individuella belöningsfunktion och hur stor vikt de bör lägga på lagets genomsnittliga belöningsfunktion. Om hyperparametern är satt till noll så delar bottarna inte med sig av belöningen de får för att spela väl. Är parametern satt till ett så delas belöningar som enskilda bottar får med resten av lagets bottar, vilket leder till att bottarna försöker se till att sina lagkamrater lyckas. [12]

	OPENAI 1V1 BOT	OPENAI FIVE
CPUs	60,000 CPU cores on Azure	128,000 preemptible CPU cores on GCP
GPUs	256 K80 GPUs on Azure	256 P100 GPUs on GCP
Experience collected	~300 years per day	~180 years per day (~900 years per day counting each hero separately)
Size of observation	~3.3 kB	~36.8 kB
Observations per second of gameplay	10	7.5
Batch size	8,388,608 observations	1,048,576 observations
Batches per minute	~20	~60

Tabell över det PPO-användande OpenAI Five systemet [12]

OpenAI Five har ett par övertag som människor inte har, såsom en reaktionstid på 80ms (det mänskliga genomsnittet i situationer som Dota ligger mellan 150 och 300 ms) och en förmåga att klicka vid den exakta bildruta den vill vilket medför att botten är ytterst skicklig på saker som att få in dödande slag på enheter.

Några upptäckter som överraskade utvecklarna är bland annat att en mer simpel form av belöningssystem gav bättre resultat än förväntat. De belönade endast agenten för att vinna, i stället för att belöna den för samtliga positiva saker den gjorde under en match. Även fast denna metod inte var lika framgångsrik som den med fler belönade handlingar, så lyckades den ändå producera en agent som presterade på en semi-professionell spelares nivå.

Utöver det så lärde sig dessutom agenten vissa koncept helt på egen hand (d.v.s. utan att få någon belöning för det specifikt). Till exempel så lärde de sig ”creep block”, vilket innebär att man blockerar sitt eget lags creeps för att få mötespunkten mellan det egna lagets och motståndarlagets creeps att hamna närmare det egna lagets tower (som i sin tur dödar creeps väldigt snabbt). [12]

OpenAI Five har också visat sig kunna samarbeta med människor på en viss nivå. [12]

6. Maskininlärning med MLP i Supreme Commander 2

Maskininlärning, närmare bestämt neuronnät, har även använts i RTS spelet Supreme Commander 2. I spelet används s.k. flerlagers perceptroner (MLP från engelskans multilayer perceptron) för att styra när bott-kontrollerade enheter ska stanna och slåss mot sina motståndare eller fly undan. [15]

6.1 Supreme Commander 2

Supreme Commander 2 har en del likt med StarCraft II men skiljer sig dock åt på ett par punkter. Spelet har mark-, luft- och sjö-enheter, medan StarCraft II endast har de två första. Samtliga spelare börjar med en stor, kraftig enhet kallad ACU som kan bygga byggnader som i sin tur bygger fler enheter. För att bygga byggnader och enheter samt underhålla dessa så behöver man tillverka/generera resurser. Utöver de två grundläggande resurserna, ”mass” och ”energy” så finns även forskningspoäng som fås från forskningsbyggnader och används till att låsa upp uppgraderingar samt nya byggnader och enheter. Målet i spelet är att förstöra fiendens ACU. [16]

6.2 MLP i spelet

Enligt utvecklarna så valdes MLP eftersom de är relativt enkla att implementera och använda. Fyra MLP användes - en för var sorts enhetstyp i spelet: land-, sjö-, bombplans- och stridsplansenheter. Detta för att varje enhetstyp skulle kunna lära sig utan att störa de andra på något sätt. En ändlig automat-användandes MLP står för bottarnas beslutstagande när det kommer till vilka motståndarenheter bottens enheter bör fokusera på, samt huruvida de bör fly när de befinner sig i ett underläge.

Indata till enhetstypernas MLP kom i form av förhållandet mellan bottens egna enheters och närliggande motståndarenheternas styrka vilket mättes i form av antal, skadekapacitet, sköldstyrka med flera, som totalt uppgick till 34 indata. Denna stora mängd indata fungerar väl i spelet i fråga men kan i många andra fall orsaka s.k. överanpassning, vilket är när en modell lär sig att fungera väl specifikt med den träningsdata som används i stället för att lära sig de mer generella mönstren som finns däri, med dålig prestanda efter träningsfasen som följd.

Utdata från spelets MLP kom i form av värden mellan noll och ett. Dessa värden representerade den förväntade nyttan av olika handlingar som var tillgängliga för enheter vid en viss tidpunkt. Ifall inget värde översteg 0.5 så tog enheten beslutet att retirera.

Till skillnad från de flesta MLP, som tränas genom att iterera genom färdiga träningsexempel, så tränades Supreme Commander 2s MLP genom att låta AIn spela mot sig själv och därmed generera träningsexempel på egen hand. När neuronnätet kom med förslag på vad bottarna skulle göra så lät man dock i stället bottarna ta ett slumpat beslut varefter man använde sig av en funktion som mätte hur bra beslutet visade sig vara. Därefter matas data in i MLP:n som justeras för att kunna ge bättre förutsägelser av handlingars nytta. Eftersom man använde slumpade handlingar när man tränade så fick modellen pröva på en stor mängd handlingar under en mångfald av omständigheter. På så vis undvek man att nätet testade samma handling under flera omständigheter och därmed så lärde sig modellen, enligt utvecklarna, snabbare och löpte mindre risk att stagnera i sitt tränande.

När en MLP lär sig så styrs storleken på nätets uppdateringar av en sorts inlärningstakts-parameter. Med en högre inlärningstakt så går träningen snabbare, dock med en ökad risk för problem såsom en extremt liten eller extremt stor gradient. En låg inlärningstakt kan däremot medföra att inlärningen tar allt för länge. I Supreme Commander 2 så har inlärningstaktsdilemmat med MLP lösts genom att inlärningstakten är hög i början men sedan avtar med tiden. Justeringarna blir alltså mindre och mindre ju närmare det optimala värdet på modellens inre variabler man kommer.

Utvecklarna av Supreme Commander 2 använde sig av även av momentum men lät parametern anta värdet noll efter träningen pågått en stund för att undvika de problem som kan uppstå då man använder parametern till en för stor grad. [15]

7. Diskussion

Bland de exempel på maskininlärd AI som tagits upp i avhandlingen så är det ett par aspekter som samtliga har gemensamt. Bland annat så har samtliga projekt använt sig av någon form av neuronnät som, i de flesta fall, använts som djupa neuronnät. LSTM-nät har varit den dominanta typen bland dessa nät. I AlphaStars och OpenAI Fives fall så har projektet överträffat åskådares förväntningar, vilket tyder på att maskininlärningsfältets utvecklingstakt har underskattats.

En av de främsta utmaningarna med maskininlärningsprojekten, men även maskininlärningsprojekt över lag, som tagits upp är att det har varit svårt att generalisera bottarnas förmågor inom vissa specifika områden till övriga områden. Ett exempel på det är att AlphaStar ursprungligen bara kunde spela som en särskild ras mot en särskild ras (Protoss mot Protoss) på en enda bana. OpenAI Five hade också en oförmåga att spela mot Dotas samtliga karaktärer.

Utöver de tekniska utmaningarna som uppstår när man försöker skapa maskininlärda bottar, så kvarstår även det faktum att en väldigt skicklig AI:s målgrupp är relativt liten. En bott av samma rang som exempelvis AlphaStar är endast önskvärd som motståndare ifall en själv tillhör de allra skickligaste spelarna. Majoriteten av datorspelare skulle högst troligtvis inte uppskatta en motståndarbott som är flera magnituder bättre än en själv.

Tillämpning av lärdomar och framsteg inom artificiell intelligens i RTS-genren är inte begränsad till enbart spelvärlden. Proteinveckning är ett område som dragit nytta av grunden som AlphaStar lagt. DeepMind byggde nämligen en proteinvecknings-AI, AlphaFold, efter att de blev klara med AlphaStar. På sistone har även självstyrda fordon och relativt autonoma robotar blivit alltmer relevanta. Dessa teknologiska gränsområden kan eventuellt dra nytta av det man lärt sig från projekt av samma slag som AlphaStar. Utöver den uppenbara likheten, d.v.s. att samtliga klassas som artificiella intelligenser, så behöver de dessutom manövrera i okända miljöer där de inte har tillgång till fullständig information om sin omgivning. Dessutom så kan de dessutom tvingas reagera på oförutsedda händelser, vilket är något som en RTS-spels AI gör med motstrategier. Ett exempel på en oförutsedd händelse är ifall en cyklist med väjningsplikt cyklar ut framför en självstyrd bil. Denna sorts dilemman är ett av de mest uppmärksammade när det kommer till etiska och säkerhetsrelaterade diskussioner kring autonoma fordon.

8. Litteraturförteckning

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville,” Deep Learning”, The MIT Press, 18 november 2016

[2] Oriol Vinyals, Igor Babuschkin, David Silver, m.fl.,” Grandmaster level in StarCraft II using multi-agent reinforcement learning”, Nature, 30 oktober 2019

[3] Bruce Geryk “A History of Real-Time Strategy Games”, ursprungligen publicerad i GameSpot, numera arkiverad vid https://web.archive.org/web/20110427052656/http://gamespot.com/gamespot/features/all/real_time/

[4] McClelland J.L., Rumelhart D.E., “Explorations in parallel distributed processing”, The MIT Press, juli 1987

[5] tappade bort källa. den kommer senare

[6] Li Deng, Dong Yu, ”Deep Learning: Methods and Applications”, publicerat av Microsoft

[7] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, ”Learning Internal Representations by Error Propagation”, MIT Press, 03 januari 1986

[8] Sepp Hochreiter, Jürgen Schmidhuber, ”Long Short-term Memory”, http://www.bioinf.jku.at/publications/older/2604.pdf , 1997

[9] Leslie P. Kaelbling, Michael L. Littman, ”Reinforcement Learning: A Survey”, Journal of Artificial Intelligence Research, 1996

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, ”Proximal Policy Optimization Algorithms”, arXiv:1707.06347 , 28 augusti 2017

[11] https://starcraft2.com/en-us/game

[12] Christopher Berner, Greg Brockman m.fl., ”Dota 2 with Large Scale Deep Reinforcement Learning”, https://arxiv.org/pdf/1912.06680.pdf, 10 mars 2021

[13] https://www.dummies.com/article/technology/information-technology/ai/machine-learning/deep-learning-and-recurrent-neural-networks-262768/

[14] https://builtin.com/data-science/recurrent-neural-networks-and-lstm

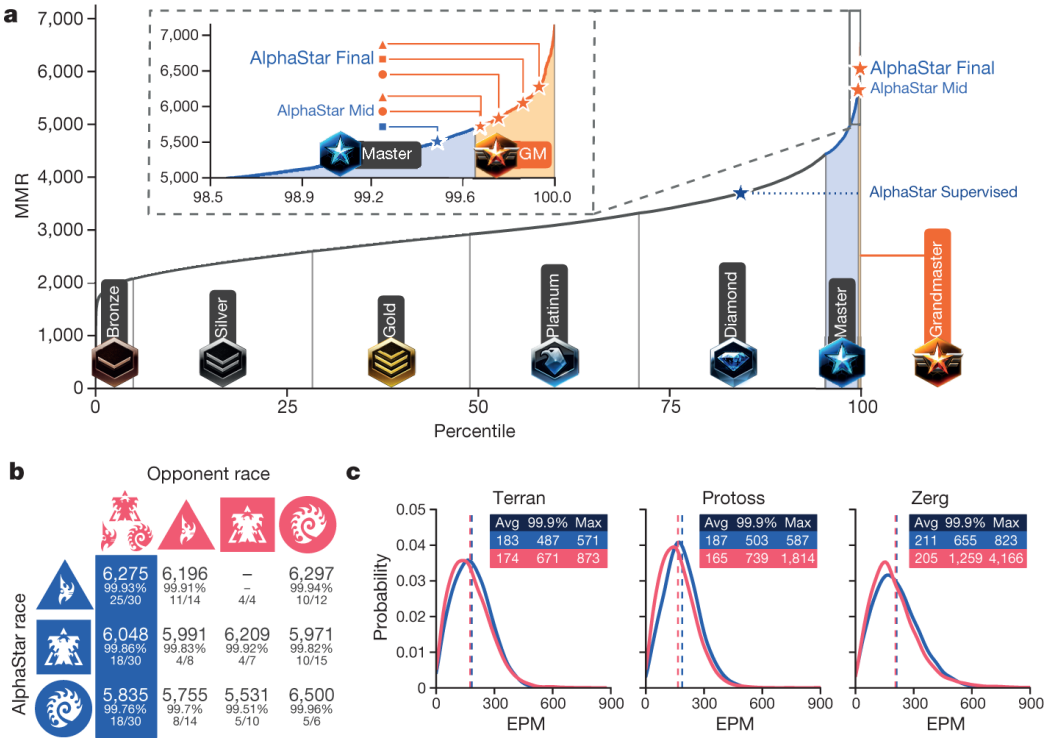
[15] Michael Robbins, ”Using Neural Networks to Control Agent Threat Response”, CRC Press, 2019

[16] https://www.ign.com/articles/2010/03/04/supreme-commander-2-review-2

[17]https://www.gamespot.com/articles/blizzcon-2008-starcraft-ii-split-into-trilogy/1100-6199172/

[18] https://www.mobygames.com/game/windows/supreme-commander-2

9. Bilagor



AlphaStar agenters MMR-fördelning per ras i spelet [2]

Spelaren ovan utför en ”creep block” i Dota 2
De två creep-grupperna kommer mötas närmare spelarens egna torn (vattnet är banans mitt)

Bild från en StarCraft II match (Protoss mot Protoss) [17]

Bild från en Supreme Commander 2 match [18]

En rätt förklarande bild på Dota 2 [12]