Forskning inom artificiell intelligens med tv-spel

ABSTRACT

Artificiell intelligens har haft en lång historia och spel har till största delen av den historien varit hur man visat hur långt forskningen har kommit. Schack var det som användes ända fram till 1997 när Deep Blue vann mot dåvarande världsmästaren i schack Garry Kasparov. 2016 vann AlphaGo mot Lee Sedol i Go ett spel många gånger mer komplext än schack. Nu har många vänt sig mot tv-spel som det nästa en dator ska tävla mot professionella spelarna i. Denna avhandling kommer titta på vad som gör spel attraktiva för forskarna, vad olika typer av spel kan träna i en artificiell intelligens och ifall det finns ett spel som skulle kunna vara standarden för testning av artificiell intelligens.

Contents

1. Inledning

1.1.Vad är artificiell intelligens

Före datorer existerade var det filosofer som tänkte på vad som skapar tankar och ifall man kunde ge ett livlöst objekt liv. Detta tankesätt utvecklades och under 1950-talet efter datorns uppfinning började forskare forska i artificiell intelligens och idén att ge livlösa objekt liv var närmare verklighet än någonsin. Artificiell intelligens är en intelligens skapad med en dator [10]. Detta fält har fortsatt forskning ännu idag men metoderna har ändrat över tiden. Under 50-talet ända tills slutet på 80-talet låg fokusen på att försöka skapa en symbolisk AI. Då detta inte ledde någonvart flyttade fokusen till att härma biologisk intelligens med tekniker såsom artificiella neuronnät och genetiska algoritmer [10]. Idag är det i huvudsak dessa tekniker som används under det övergripande begreppet maskininlärning vilket kommer vara fokusen i denna avhandling.

1.2. Artificiell generell intelligens

Artificiell generell intelligens (AGI), också kallad stark artificiell intelligens, är en AI som klarar av många olika uppgifter i olika miljöer utan extra träning [2]. Detta är AI såsom den är avbildad i populärkulturen, en intelligens som är oskiljbar från den mänskliga intelligensen, Alan Turing tänkte redan 1950 på en sådan AI då han la fram det nu berömda Turingtestet[19].

En artificiell generell intelligens är ett mål av många AI-forskare och det finns flera organisationer som har pågående AGI-projekt, bland dessa är de största DeepMind, OpenAI och Human Brain Project [3]. Både OpenAI och DeepMind är oroade över farorna med AI och har satsat mycket resurser på att en potentiell AI ska vara säker, de har även samarbetat för att nå detta mål [3], [12], [24].

Många forskare och andra inom teknikfältet har uttryckt oro över hur en AI utan tillräckliga säkerhetsåtgärder skulle vara en fara för hela mänskligheten, bland de som har uttryckt sig är Elon Musk en av grundarna av OpenAI. Det finns olika teorier över hur en AI skulle kunna bli en fara men den vanligaste teorin är en

superintelligens. En superintelligens skapas genom rekursiv självförbättring, intelligensen skriver om sin egen programvara till en bättre version och den nya versionen skulle kunna förbättra sig själv ännu mer och varje version kan förbättra sig själv snabbare och snabbare [22].

2. Olika typer av artificiell intelligens

Forskningen inom AI började redan på 40-talet med försök att simulera hjärnan med cybernetik. Efter att datorn blev tillgänglig på 50-talet började forskningen inom AI ta fart. Till att börja med försökte forskarna använda symbolisk artificiell intelligens. Denna typ av AI har i stort sett blivit övergiven och ersatt av vad som idag kallas maskininlärning.

2.1. Symbolisk artificiell intelligens

Symbolisk artificiell intelligens kallades den teknik som användes från 1950-talet till slutet på 1980-talet. En symbolisk AI fungerade genom att forskare byggde upp en kunskapsdatabas där intelligensen kunde söka upp information för att upptäcka ny information. För att söka upp informationen skulle forskarna skapa en algoritm vilket skulle fungera som hjärnan i intelligensen.

Denna teknik visade sig fungera för att lösa simplare problem som att förstå enkla meningar, men att göra något mer komplicerat så som att förstå ett språk visade sig omöjligt [10]. Forskningen i symbolisk artificiell intelligens höll på länge och även inom denna metod fanns olika versioner.

2.2. Maskininlärning

Ethem Alpaydin skriver i boken Introduction to Machine Learning: ”Maskininlärning använder exempeldata eller tidigare erfarenhet för att optimera ett prestandakriterium” [1]. Maskininlärning används redan i stora delar av dagen samhälle, det används för att sortera skräppost i e-postklienten, att automatiskt känna igen ansikten på Facebook, och på många andra sätt. Sådan användning av maskininlärning kallas svag artificiell intelligens och är motsatsen till en artificiell generell intelligens. En svag artificiell intelligens används endast för en eller några specifika uppgifter och kan inte jämföras med den mänskliga intelligensen.

Maskininlärning delas ofta in i tre breda kategorier, övervakat lärande, oövervakat lärande och förstärkningslärande (reinforcement learning) [1].

2.2.1. Övervakat lärande

Övervakat lärande ger programmet indata och utdata och det är sedan maskinens uppgift att hitta en länk mellan indatan och utdatan. Detta kan användas t.ex. för att sortera skräppost. Indata i detta exempel är e-posten och utdata helt enkelt ifall ett mejl är skräppost eller ej. Programmet lär sig vad som skiljer skräppost från ett vanligt mejl genom att jämföra en stor mängd skräppost med riktig e-post där den får veta vilka som är vilka. Efter att ha analyserat alla mejl har programmet skapat några kriterier över vad som är skräppost och vad som är riktig e-post. Programmet kan då analysera inkommande e-post och jämföra den mot alla kriterier det lärde sig från träningsstadiet och vart efter den lär sig ny information kan programmet uppdatera kriterierna [1].

2.2.2. Oövervakat lärande

Oövervakat lärande ger programmet endast indata och det är då maskinens uppgift att hitta en länk mellan alla indata. Detta kan användas för att gruppera t.ex. dokument i olika kategorier [1].

2.2.3. Förstärkningslärande

Förstärkningslärande används då datorprogrammet måste nå ett mål genom en serie av beslut. I detta fall är enskilda beslut irrelevanta och det är endast helheten som bedöms. Exempel på när förstärkningslärande kan användas är ett program som spelar schack. I schack är enskilda drag oftast inte bra eller dåliga utan det är helheten av ett parti schack [1].

2.2.4. Djup maskininlärning

Djup maskininlärning (deep learning) är inte en skild kategori av maskininlärning som övervakat, oövervakat och förstärkningslärande är; djup maskininlärning är en mer avancerad typ av maskininlärning ofta baserad på ett artificiellt neuronnät. Liksom artificiella neuronnät använder djup maskininlärning flera gömda lager,

varje lagers utdata blir till nästa lagers indata. Varje lager hjälper att bygga upp den information som behövs [7].

Djup förstärkningslärande har blivit ett populärt forskningsområde inom artificiell intelligens, många av de största framgångssagorna under de senaste åren har använt sig av denna metod. Djup förstärkningslärande är för tillfället den mest lovande typen av maskininlärning för att skapa en AGI [2]. DeepMind och OpenAI fokuserar på djup förstärkningslärande i sin forskning.

2.2.5. Övrigt

Artificiella neuronnät (ANN) tar inspiration från biologiska neuronnät. ANN försöker simulera biologiska neuronnät som hjärnan, på grund av hjärnans förmåga att enkelt lära sig sådant som det är mycket svårt att lära en dator. Exempelvis att förstå tal, känna igen ansikten o.s.v. Forskare tror att ifall de förstår hur hjärnan klarar av dessa saker så kan de utforma en lösning för att lära en dator att göra samma sak [1]. Ett ANN använder flera lager av neuroner. Det första lagret får indata, dessa data går igenom flera gömda lager av neuroner och varje neuron är länkad till varje neuron i nästa lager, varje länk har också en vikt som ökar eller sänker signalen som går över en länk. Det sista lagret av neuroner ger utdata [21].

Genetiska algoritmer eller genetisk programmering är inspirerad av biologisk evolution. Först ger algoritmen eller datorprogrammet ut några slumpmässigt valda lösningar, de bästa av dessa lösningarna ”parar sig” för att kombineras till förhoppningsvis en ännu bättre lösning, de bästa lösningarna från de parade lösningarna får i sin tur para sig och detta upprepas tills ett acceptabelt svar eller maxantalet tillåtna generationer har nåtts. Emellanåt läggs en muterad lösning in i bland lösningarna, detta är en lösning som har blivit slumpmässigt ändrad [9]. Mutationen leder ofta till en effektivare lösning [9].

Monte Carlo Tree Search (MCTS) är en sökalgoritm som kan användas som AI i spel. Genom att först välja de drag som är mest lovande, sedan lägga till en ny nod då algoritmen når slutet och tillslut ändra vikterna på de noder som använts beroende på hur matchen slutar, bygger algoritmen upp ett träd som har de mest lovande dragen vid olika tillfällen i en match [5]

Djup Q-inlärning är ett djupt förstärkningslärandeprogram som använder djup manskininlärning i samband med Q-inlärningstekniken. Djup Q-inlärning är skapat av DeepMind [11].

3. Maskininlärning med spel

Så länge som människor har forskat inom artificiell intelligens har spel använts för att testa hur avancerad en AI är. Redan 1951 skrev Christopher Strachey ett program för att spela damspel och några senare månader hade Dietrich Prinz skrivit ett schackprogram. Dessa program var simpla och inte snabba (Prinz program spelade inte ett helt parti av schack och löste endast problem som kunde vinnas inom ett par drag [23]) men deltog i att spel och AI-utvecklingen gick hand i hand. Stracheys program utvecklades under 50- och 60-talet av Arthur Samuel till ett program som kunde utmana en skicklig amatör i damspel [15].

Schack blev standardtestet som användes för att testa hur avancerad en AI var fram till 1997, då datorn Deep Blue skapad av Ibm slog dåvarande världsmästaren i schack Garry Kasparov. AlphaGo är ett datorprogram som skapades av DeepMind för att spela go, ett spel med många gånger mer möjliga drag än schack. AlphaGo blev det första datorprogrammet att vinna mot en professionell go-spelare 2015, och 2017 vann AlphaGo mot den bästa spelaren i go [18]. Nu har många forskare börjat använda tv-spel istället för brädspel för att testa AI. DeepMind har samarbete med spelutvecklarna av strategispelet StarCraft II vid Blizzard Entertainment, tillsammans skapade de ett programmeringsgränssnitt (api) som tillät en extern AI att styra spelet. OpenAI har skapat en AI som spelar Dota 2. Det finns även ramverk för AI att spela b.la. Atari 2600-spel och Doom.

3.1. AI i tv-spel

Tv-spel har länge använt en eller annan form av AI som motståndare. Redan på 1970-talet började arkadspel implementera simpla AI, dessa tidiga AI följde ett enkelt mönster och har väldigt lite i samband med AGI-forskningen. Under 80- och 90-talet då AGI-forskningen stod mer eller mindre stilla gick spel AI snabbt framåt. Från de enkla mönstren som i Space Invaders, till mer komplicerade och varierande mönster bara ett år senare i Galaxian.

Under 80-talet var det inte bara de kraftfulla arkadmaskinerna som klarade av den då mer komplicerade AI:n, hemkonsoler kunde nu också ha inte bara ha statiska mönster utan en AI som delvis reagerade på vad spelaren gjorde. På 90-talet fick spelare helt nya utmaningar i strategispel med finita tillståndsmaskiner (finite state machines) som gjorde det möjligt för AI:n att göra beslut och planera [17].

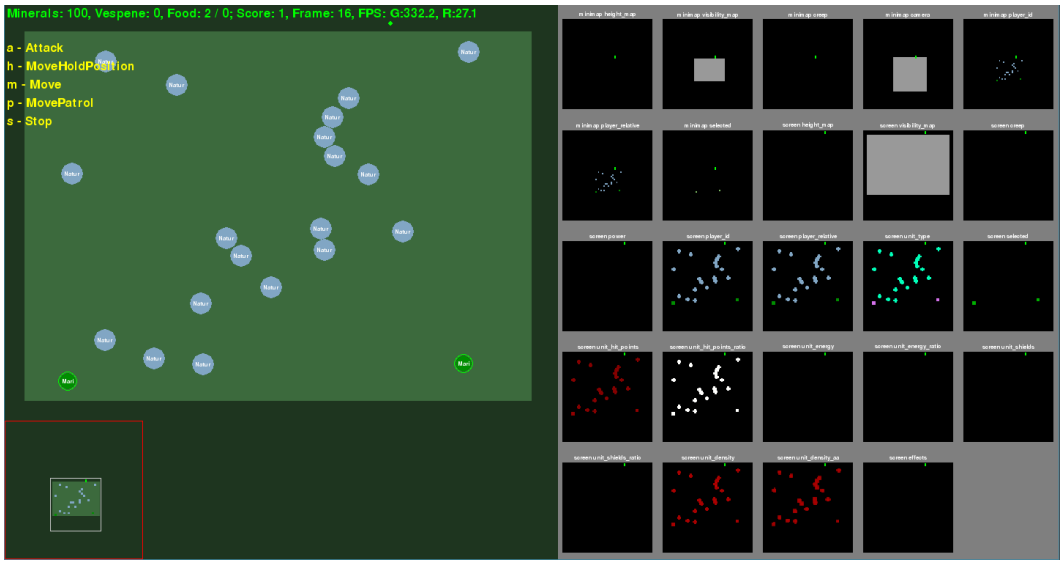
Under det tidiga 2000-talet utvecklades flera spel med AI baserat på maskininlärning, Black & White från 2001 hade en AI-varelse som kan ändra sitt beteende beroende på hur spelaren belönar och bestraffar den. AI i spel har i största allmänhet inget behov av att vara så skicklig som AlphaGo på ett spel eftersom ett spels AI har i uppgift att skapa en utmaning som spelaren kan slå. En AI i ett spel behöver inte heller vara så komplicerad som en AGI eftersom den endast har ett få antal möjliga funktioner tillgängliga. En AGI är inte kontrollerbar till den nivå som krävs för att skapa en underhållande upplevelse. Att köra en så komplicerad AI kräver alldeles för mycket processorkraft för att vara realistiskt att köra på en persondator.

3.2. StarCraft 2

Starcraft 2 är ett realtidsstrategispel (RTS) som släpptes 2010 och är utvecklat av Blizzard Entertainment. I Starcraft 2 kontrollerar spelaren en av tre raser, Terran, Zerg eller Protoss, målet med spelet är att samla resurser för att bygga upp en bas och en armé, och med den armén förstöra fiendens bas. Spelet spelas oftast i flerspelarläget i en mot en (1v1) läget. Spelet har en inbyggd AI men den utgör ingen större utmaning för en skickligare Starcraft 2 spelare och på högre svårighetsgrader fuskar den b.la. genom att få extra resurser och alltid ha syn på vad motståndaren gör. De flesta realtidsstrategispel så som Starcraft kräver att en spelare utforskar kartan för att ta reda på vad motspelaren gör, och endast den del av kartan som spelarens enheter ser uppdateras i realtid, spelarens kamera måste också flyttas runt. Detta gör att det är möjligt att gömma byggnader och enheter från motspelaren och man måste utforska kartan för att ta reda på vilken strategi som motspelaren använder.

3.2.1. AI-forskning i StarCraft 2

DeepMind i samarbete med Blizzard utvecklade SC2LE (StarCraft II Learning Environment) som tillåter en extern AI kontrollera spelet. Detta sker genom ett api som skapades av Blizzard för Starcraft 2 och ett Python startskript (wrapper) PySC2 skapat av DeepMind. PySC2 definierar vad en spelare kan göra i spelet, det låter även den artificiella intelligensen observera spelet. Programmeringsgränssnittet skapar en vy av spelet delat i olika lager, se figur 1, ett skilt lager för information så som var på kartan spelarens kamera är centrerad, höjden på olika delar av kartan, vilken enhet som är vald o.s.v. En mänsklig spelar ser all denna information via den tredimensionella grafiken och användargränssnittet i spelet, se figur 2. För att göra de lättare att studera vad agenten gör skapar PySC2 en vy för att visa vad den ser, där finns både en bild som liknar vad en mänsklig spelare skulle se i spelet med alla lager visade på en gång och vad agenten använder alla lager separatm, också i figur 1, i framtiden vill DeepMind att agenten skall se den ihopsatta bilden för att vara närmare till hur en mänsklig spelare skulle se spelet [20].



Figur 1. Vyn skapad av PySC2. Till höger är den olika vyerna en agent använder för att se spelet. Till vänster är den sammansatta vyn samt en lista med alla tillgängliga aktioner för de valda enheterna.

Figur 2. Standard vyn i Starcraft 2

Realtidsstrategisspel så som Starcraft 2 har många fördelar för forskning. De kräver att en spelare kontrollerar en stor mängd enheter med olika användningsområden, det finns arbetare som samlar resurser, olika typer av soldaterenheter som är kraftfulla mot en del andra soldaterenheter och svaga mot andra. Det krävs att spelaren bygger upp en armé med olika typer av enheter som kan kontra det fienden bygger. Utöver att ha byggt en armé så spelar placeringen av varje enhet i armén roll, inte bara före en strid men under en strid kan spelare vara tvungna att flytta en enhet till en bättre position. En professionell Starcraft 2 spelare har ofta över i medeltal över 300 knapptryck per minut (actions per minute eller APM) detta är något som en dator lätt kan överskrida men i PySC2 är det möjligt att välja hur ofta agenten kan agera, som standard i DeepMinds testning var att tillåta agenten att agera var åttonde bild (frame) vilket blir ungefär samma som 180 APM [20]. Utöver att kontrollera spelet, behövs också framtidsplanering som i schack, men med ofullständig information. Alltså måste en spelare göra en plan och anpassa den baserat på information man lär sig allt efter spelet utspelar sig. Kanske motståndaren bygger enheter som kontrar spelarens enheter, då krävs det att spelaren bygger om sin bas och producerar nya enheter ifall det finns resurser och tid för detta, annars kan det vara möjligt att vinna ett slag genom att styra enheterna bättre än motståndaren, det kan även gå att lura motståndaren att bygga om sin bas

genom att producera ett fåtal av en enhet som kontrar motståndarens enheter och låta dem bli upptäckta. Som man ser i dessa exempel är de möjliga dragen i ett realtidsstrategispel många fler än de som man har tillgängliga i schack. För sitt experiment skapade DeepMind ett antal kartor som testade olika av de skickligheterna som behövs för att spela Starcraft 2, dessa testade allt från att flytta en grupp med soldater till en specifik plats på kartan till mer komplicerade uppgifter så som att vinna en strid mot en grupp med fiender som har övertaget. Dessa tester klarade en förstärkninginlärd agent av lika bra som en skicklig mänsklig spelare, men en likadan agent kunde inte vinna en riktig match mot den lättaste av spelets egna AI.

Det är svårt att belöna agenten i en match av Starcraft 2. Det simplaste sättet att belöna skulle vara att ge 1 poäng ifall agenten vinner, -1 ifall den förlorar och 0 ifall det blir oavgjort. Denna belöningsstrategi gav inga lovande resultat, en match har för många variabler för att en agent skall kunna lära sig vad som gjorde att den vann eller förlorade. En annan belöningsstrategi som användes var spelets inbyggda poängsättning. Starcraft 2 ger poäng baserat på hur samlade resurser, utvecklade uppgraderingar, och enheter som är vid liv; poängen minskar då en enhet eller byggnad förstörs [20]. Denna belöningsstrategi ledde till att agenten bara samlade resurser och inte byggde några enheter, då kunde den inte förlora några enheter och förlora poäng. Hur man belönar en agent kan ha stora ändringar på hur den spelar. För att agenten ska på bästa sätt ska lära sig spela självständigt krävs ett enkelt system för att ge belöningar och bestraffningar, den första strategi skulle därför vara den optimala strategin för en helt självständig AI och är den DeepMind stävar efter att använda. Med ett system som ger poäng för att samla resurser och tar bort poäng för att förlora enheter, leder det som tidigare till att agenten är rädd för att bygga enheter ifall den skulle förlora dem. Fokusen borde ligga på något annat istället, ifall man belönar agenten för varje resurs spenderad istället för varje resurs tjänad skulle detta kunna leda till att agenten spenderar de resurser den tjänar och bygger nya enheter och byggnader. Att bestraffa agenten för att förlora en enhet är inte helt fel men det kan leda till att den är rädd för att gå i strid med fienden, så att belöna agenten för varje fiendeenhet som den förstör skulle också vara viktigt. DeepMind hade en liknande belöningsstrategi på en av deras testkartor. På denna karta belönas agenten varje gång den förstör en fiendeenhet och bestraffas varje

gång den förlorar en egen, ifall agenten förlorar alla enheter bestraffas den ytterligare och ifall den förstör alla fiender belönas den ytterligare. Hur man belönar agenten är en del som gör det svårt att använda realtidsstrategispel för att träna en förstärkninginlärd artificiell intelligens, det finns många strategier var man inte bygger upp en stor armé och därav inte spenderar många resurser vilket är en strategi en agent inte skulle lära sig med min tidigare utlagda belöningsstrategi.

Ett annat sätt som PySc2 kan användas för att träna en artificiell intelligens är genom att studera repriser av matcher, detta är ett vanligt sätt även för professionella spelare att lära sig både av sina misstag och av vad de gjorde bra i en match. Starcraft 2 har fritt tillgängliga många hundratusen repriser av spelare på olika skicklighetsnivåer, DeepMind använde 800 000 för att träna en agent att både förutse vem som kommer vinna i reprisen och även förutse vad en spelare kommer att göra som nästa i en repris. Att förutse vem som kommer att vinna en match är från början 50% eftersom varenda ras är lika starka, redan efter några minuter hade den bästa agenten en 64% chans att förutse rätt, detta ökade ytterligare vart efter matchen fortsatte. Denna träning hjälper agenten lära sig vad som gör att en spelare vinner eller förlorar, vilket kan användas då den endast blir belönad av att vinna en match eftersom den vet bättre vad som gjorde att den vann eller förlorade. Träningen att förutse vad som en spelare kommer att göra näst hjälpte agenten att spela bättre, efter att ha tränat med detta byggde den flera enheter och göra bättre beslut. Denna träning var gjort med övervakat lärande istället för endast med förstärkningslärande som tidigare. Forskarna vid DeepMind tror att RTS är den bästa genren av spel för att träna en AI idag på grund av hur många olika skickligheter det kräv för att vara bra på ett sådant. Starcraft 2 används för att det har den största mängden spelare och en stark e-sportscen med många professionella spelare.

3.3. Dota 2

Dota 2 är ett aktionrealtidsstrategispel (MOBA) en undergenre till RTS. Dota 2 utvecklades av Valve Corporation och släpptes 2013. I Dota 2 väljer spelaren en hjälte bland de över 100 tillgängliga, varje hjälte har en egen spelstil med olika krafter som den kan använda. Till skillnad från ett RTS spelas ett MOBA för det mesta endast en karta, denna karta har tre filer med torn och datorstyrda enheter

från båda lagen som marscherar mot sin fiendesbas. Ett MOBA spelas oftast i ett fem mot fem läge var varje spelare styr en egen hjälte. Målet med spelet är förstöra fiendebasen, men för att göra detta måste hjältarna förstöra alla torn i åtminstone en fil. Under spelets gång är det möjligt att uppgradera sin hjältes krafter genom att få nya nivåer, det är också möjligt att köpa utrustning och andra hjälpmedel för att öka sin hjältes förmågor. För att köpa utrustningen krävs guld, och för att få högre nivåer krävs erfarenhet (XP), som en spelare tjänar genom att döda fiendens enheter, hjältar och torn, det finns också möjlighet att gå utanför kartans filer för att döda neutrala monster i den så kallade djungeln. De skickligaste spelarna kan få extra guld genom att vara den sista som slår en fiende, och det är också möjligt att förneka fienden det sista slaget genom att få det sista slaget på en vänligenhet.

3.3.1. AI-forskning i Dota 2

Vid OpenAI har forskarna skapat en AI för att spela Dota 2. Denna AI spelar ännu inte 5v5-läget, forskarna har fokuserat på att lära AI:n att spela 1v1-läget först. 1v1-läget fungerar i stort sett likadant som 5v5-läget med de skillnaderna att spelarna kan endast använda filen i mitten av kartan och målet är endast att förstöra ett fiendetorn eller döda motståndarens hjälte två gånger. Idén med att använda detta spelläge istället för standardläget var samma som DeepMind hade med testkartorna i Starcraft 2, en förenklad version av det fulla spelet där agenten kunde lära sig spela spelet med en mindre mängd val. En 1v1-match tar också betydligt kortare tid att slutföra så agenten hinner träna i flera matcher. För att fokusera träningen på att lära agenten hur man spelar låste även OpenAI agenten till en hjälte och lät den endast välja mellan ett tiotal olika saker ur utrustningslista. Deras agent lärde sig spela Dota 2 helt genom att spela mot sig själv. Detta träningssätt gav resultat snabbt och inom några månader gick AI:n från att inte kunna spela Dota 2 till att vinna i 1v1-läget mot de bästa professionella Dota 2 spelarna i världen. OpenAI vill fortsätta forskningen och i framtiden har ett lag med agenter spela i 5v5-läge.

Till skillnad från Starcraft 2-agenten skapad av DeepMind använder sig inte Dota 2-agenten av en visuell vy för att se vad som händer i spelet. Dota 2 har ett programmeringsgränssnitt för att skapa botskript vilket kan användas för att göra allt en spelare kan göra, denna api ger också botten all information som skulle vara

tillgänglig för en spelare. Detta är hur OpenAI skapade grunden av sin Dota 2-bot som deras agent styr [6]. Medans forskarna vid DeepMind försöker ha sin agent att uppleva spelet så likt en mänskligspelare som möjligt är forskarna vid OpenAI mer fokuserade på att få en agent som är bra på Dota 2.

Dota 2-agenten är en förstärkningsinlärd bot som tränat mot sig själv. Belöning ges då agenten vinner och får det sista slaget på en fiendeenhet, även hur mycket liv hjälten har kvar kan bestraffas eller belönas. OpenAI har finjusterat hur agenten spelar under tiden den har tränat, delvis på grund av begränsningar av att endast träna mot sig själv. En del av de professionella spelarna hittade sätt att utnyttja strategier agenten aldrig sett tidigare för att vinna en match. Bland annat Per Anders Olsson (Pajkatt) som använde utrustning för att öka sin hjältes liv på ett sätt agenten aldrig sett vilket ledde till att Pajkatt kunde vinna matchen. OpenAI tillät sedan agenten att använda denna utrustning för att lära sig hur den skulle kunna undvika att göra samma misstag i framtiden. Då botten skulle visas upp under The International 2017, e-sportsturneringen med den största prispotten någonsin [14], använde botten en strategi där den låtsades vara en mycket sämre bot och lurade motspelaren att attackera, en strategi som fungerade några gånger men efter att spelaren lärde sig vad som hände inte fungerade alls. För att hinna i tid till The International med en fungerande bot, tog de en tidigare agent som hade en bra strategi i början av matchen men inte var lika skicklig som den senare agenten efter det. Mitt under matchen böt de agent för att ha en bot som kunde slå Danil Ishutin (Dendi) på scen.

Dota 2 och Starcraft 2 kräver många av de samma egenskaperna för att spela bra, framtidsplanering i hur man uppgraderar sin hjälte och vad man skaffar för utrustning i, översikt över vad som händer i de olika filerna och kontroll över sin hjälte, men det finns några skillnader. Den största skillnaden mellan Dota 2 och Starcraft 2 är mängden enheter en spelare kontrollerar, i Dota 2 styr en spelare en hjälte istället för en hel armé i Starcraft 2, istället krävs samarbete mellan 5 spelare i Dota 2 (i standardläget 5v5) för att ha en chans att vinna. OpenAI har som mål att kunna få ett lag med både mänskliga- och artificiellaspelare. Effektiv kommunikation mellan lagmedlemmar är ett måste för att vara ett bra lag, inbyggda verktyg i spelet som att markera på kartan var en motståndare kan vara o.d. Dessa verktyg kan inte ersätta mer effektiva sätt att kommunicera, för tillfället är det

effektivaste sättet att kommunicera bland mänskor tal. Detta skulle förstås vara ett stort hinder för ett blandat lag Dota 2 lag. OpenAI valde inte spelet på grund av vilka egenskaper det skulle testa i en agent, i en intervju sa de att de tittade på den populära strömningswebbsidan Twitch.tv efter vilket spel som var mest populärt och mötte deras kriterier. Dessa kriterier var att det finns ett inbyggt programmeringsgränssnitt för att låta en AI styra spelet, en aktiv professionell scen och det finns många repriser tillgängliga att analysera [6].

För att börja träna en AI att spela 5v5-läget så tänker OpenAI använda av de stora mängderna repriser som är tillgängliga. Genom att låta en agent studera dessa repriser och med beteendekloning kan de få agenten att spela på ett mer mänskligt sätt.

3.4. Atari 2600

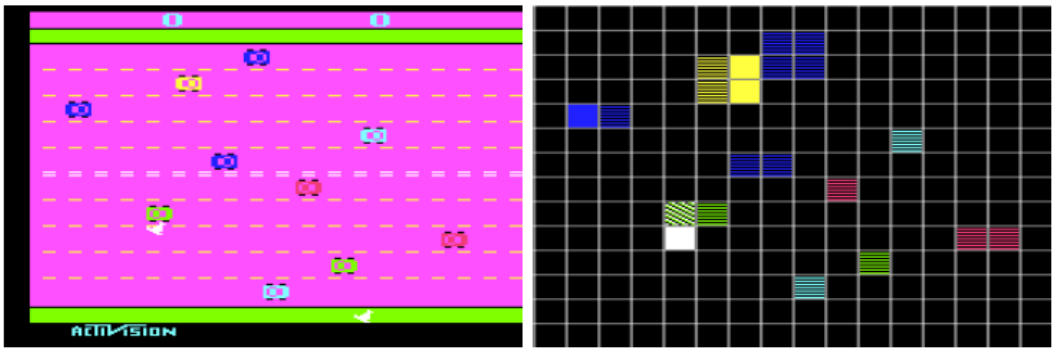
Atari 2600 är en spelkonsol producerad av Atari som släpptes 1977. Atari 2600 var en av de första spelkonsolerna som hade spel på spelkassetter och kunde därför spela många olika spel istället för endast de som var inbyggda i konsolen. Det är delvis tack vare Atari 2600 som tv-spelsbranschen blev så populär under 80-talet. Målet i de flesta spelen va endast att få så höga poäng som möjligt.

3.4.1. AI-forskning med Atari 2600-spel

Atari 2600-spel har använts av flera olika forskningsgrupper för att träna en AI. The Arcade Learning Environment (ALE) är ett ramverk utvecklat så det ska vara lättare att utveckla förstärkningslärande agenter för att spela Atari 2600-spel [4]. ALE baserar sig på en Atari 2600-emulator för att köra och styra spelen, ALE ser också hur mycket poäng en agent har i ett spel och ifall spelet är över. På grund av hur mycket mer kraftfulla datorer är idag jämfört med Atari 2600-konsolen kan spelen spelas med 6000 bilder per sekund, hundra gånger snabbare än de 60 bilder per sekund som konsolen var skapad för, detta tillåter mycket snabbare träning av en agent. Utvecklarna bakom ALE anser att Atari 2600-spela är ett bra sätt att träna en AI eftersom även om spelen är simpla, den stora mängden spel tillgängliga och variationen är ändå ett bra steg mot en mer avancerad AGI.

Utvecklarna av ALE tränade fem olika agenter i fem tusen träningspass (ett pass pågår tills spelet är över eller i 5 minuter, 18 000 bilder) och testade sedan dessa

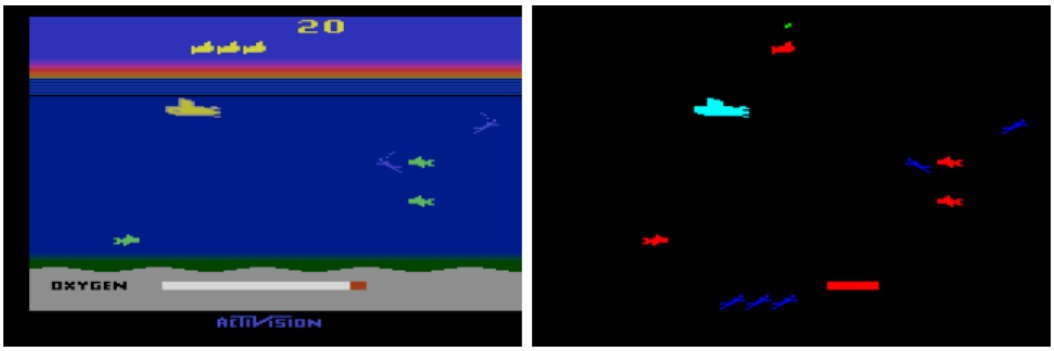
agenter i fem hundra utvärderingspass i fem olika Atari 2600-spel. Dessa agenter jämfördes med hur väl ett antal baslinje agenter och en mänsklig nybörjare klarade av att spela dessa spel. I alla fem spelen var det en av de tränade agenterna som klarade sig bäst, men ingen av agenterna var överlägset bättre än de andra. Av de fem agenterna använder 4 visuelldata för att analysera informationen i spelet, den femte metoden, kallad RAM analyserar det data som finns i konsolens 1024 bitar av arbetsminne. I fortsatt testning med ett större antal spel visade det sig att metoden kallad BASS (Basic Abstraction of the ScreenShots) oftast klarade sig bäst och att metoden DISCO (Detecting Instances of Classes of Objects) klarade sig sämst. BASS ytterligare simplifierar spelens grafik, genom att visualisera spelet på en svart bakgrund och endast använda kvadrater på ett rutmönster, i figur 3 är denna metod använd i spelet Freeway i SECAM läget vilket endast använder 8 färger istället för de 128 färgerna tillgängliga i andra lägen som NTSC. DISCO har en liknande idé men simplifierar inte grafiken lika mycket, färgerna är också bestämda beroende på objekt typen och inte de färger objekten har i spelet se figur 4, i spel som DISCO-agenten inte tränat på ledde detta till att den krävde för mycket prestanda för att göra ett snabbt beslut i spelet. För dessa tester krävdes även att ett antal olika sätt att räkna poängen testades eftersom olika spel gav poäng på olika sätt och dessa agenter tränades på många olika spel. Några olika normaliseringsalgoritmer testades för att få de olika poängen till samma skala, efter att poängen normaliserades så måste poängen från de olika spelen aggregeras för att se hur agenterna klarade sig över all spelen [4].



Figur 3. Till höger är den BASS-simplifierade versionen av spelet Freeway till vänster [4].

Andra forskar har också använt ALE för sin forskning. DeepMind använde ALE för att träna en agent med djup Q-inlärning (DQN). Djup Q-inlärning använder ett faltningsneuronnät (convolutional neural network, CNN). I dessa tester hade

agenten som tidigare endast visuell information tillgänglig, och för att ha samma miljö stängdes ljudet av för den mänskliga professionella speltestaren. Belöningsstrategi som användes ändrades dock, varje gång agenten gjorde något som gav poäng i spelet fick den endast 1 poäng som belöning, och då den gjorde något som tog bort poäng i spelet togs 1 poäng bort, på detta sätt hade alla spel samma poängskala. Nackdelen med denna belöningsstrategi är att det inte finns någon skillnad på hur bra en något som blir belönat var, eftersom all belöning hade samma värde. Med denna metod kunde forskarna få agenten att spela lika bra eller bättre än den professionella spelaren i den största delen av de 49 testade spelen, agenten hittade även långsiktiga strategier i en del av spelen. [11]



Figur 4. Spelet Seaquest och till vänster den simplifierade DISCO-grafiken, röda objekt är fiender, blåa människor som skall räddas av spelaren i cyan [4].

Atari 2600-spel fastän simpla är varierande vilket är väldigt nyttigt för AGI-forskning då många olika färdigheter tränas och testas. Över 100 spel släpptes av Atari mellan 70- och 90-talet och hundratals flera släpptes av tredje parter, ännu idag släpper entusiaster hemgjorda spel, med så många spel finns det nästan inget slut på möjligheterna att träna en AI, att få spelen att köra på ALE kräver dock möjligtvis extra arbete, eftersom ett spel måst fungera på emulatorn som används av ALE. Men även om det finns hundratalsspel att välja bland är grafiken på konsolen och den begränsade kontrollmöjligheten ändå en hindrande faktor och endast en språngbräda till något bättre.

3.5. DOOM

Doom (ofta skrivet DOOM) är ett förstapersonsskjutarspel (FPS) utvecklat av id Software som släpptes 1993. Efter att ha släppt Wolfenstein 3D år 1992, utvecklade id Software Doom för att ha snabbare spelstil och ha bättre grafik.

Kartorna i spelet är också mer komplicerade än de i Wolfenstein. Doom gjorde att FPS blev en populär genre under 90-talet och ännu idag är de populäraste spelen ofta FPS. I Doom styr spelaren en namnlös karaktär ofta kallad ”Doomguy” på Mars efter att demoner har invaderat. I FPS ser spelaren från karaktärens ögon och styr den direkt, oftast har spelaren ett antal olika vapen att döda fiender med. Doom har utöver enspelarläget, ett flerspelarläge där upp till fyra spelare kan spela mot varandra (ett större antal spelare tilläts i senare utgåvor av spelet). För att vara bra på Doom krävs det att man har en snabb reaktionsförmåga då man ser en fiende, men också att man är bra på att sikta. Det är möjligt att undvika många av projektilerna i Doom p.g.a. spelarens hastighet, detta är extremt viktigt i de svårare enspelarkartorna och i flerspelarläget. Ett gott lokalsinne och uppmärksamhet i 360-grader är också viktigt.

3.5.1. AI-forskning i Doom

Projektet ViZDoom skapades som en forskningsplattform för visuellt förstärkningslärande. Utvecklarna bakom ViZDoom ansåg att den tvådimensionella världen i Atari 2600-spel har flera nackdelar jämfört med en mer realistisk tredimensionell värld så som i FPS som Doom. Atari 2600-spel använder ett tredjepersonsperspektiv i sina spel, alltså man ser karaktären från utsida och styr den därifrån, eftersom man styr Doom från ett förstapersonsperspektiv skulle detta vara mer enligt hur en robot skulle uppleva den verkliga världen. Agenterna som tränats på Atari 2600-spel är redan i de flesta fall bättre än en mänsklig spelare och lär sig spelen snabbt, så mer utmanande problem behövs.

Doom valdes som grund för ViZDoom på grund av ett antal kriterier, b.la. öppen källkod, flerspelarläge, hur lätt det är att skapa egna kartor. Moderna datorer är också så mycket mer kraftfulla än de som spelet skapades för vilket gör det möjligt att spela spelet i nästan 7000 bilder per sekund jämfört med de 35 bilder per sekund spelet vanligtvis kör i. Utvecklarna skapade två testkartor för att träna agenter, den första var ett enkelt rum där agentens uppgift var att skjuta monstret på andra sidan av rummet så snabbt som möjligt, på den andra kartan var uppgiften att överleva på en karta där golvet var gjort av gift som skadade spelaren, för att överleva måste spelaren plocka upp paket som ger mera liv [8]. Möjligheten att lätt skapa dessa kartor

ViZDoom har använts för en tävling vid konferensen CIG (Conference on Computational Intelligence in Games) där de tävlande lämnade in en agent som sedan spelade ett antal dödsmatcher emot varandra.

3.6.Övrigt

Det största problemet med alla de tidigare nämnda artificiella intelligenserna är att de är fokuserade på ett spel, eller ett system, även med specialgjorda kartor kan det vara en begränsning. Ett tv-spels beskrivningsspråk (Video Game Description Language, VGDL) skapades för att beskriva ett tvådimensionellt spel för en agent. [16] Ett ramverk General Video Game AI (GVGAI) skapades för tävlingen General Video Game AI Competition. Detta ramverk skrivet i java låter en agent spela alla spel beskrivna av VGDL. Tillskillnad från många av de andra agenterna jag tagit upp använder detta ramverk inte en visuell analys av spelet, ramverket ger agenten information om spelets läge via ett javaobjekt. Agenten får inte veta spelets regler, hur olika objekt i spelet reagerar med varandra eller hur man vinner, detta är upp till agenten att ta reda på [13]. Det är möjligt att det finns fler möjligheter till variation inom spel beskrivna med VGDL, och det är definitivt lättare att implementera än de som finns tillgängliga via ALE.

4. Avslutning

Källförteckning

[1] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2010.

[2] P.-A. Andersen, ”Deep Reinforcement Learning using Capsules in Advanced Game Environments”, *arXiv [cs.AI]*, 29-jan-2018.

[3] S. Baum, ”A Survey of Artificial General Intelligence Projects for Ethics, Risk, and Policy”, nov. 2017.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, och M. Bowling, ”The Arcade Learning Environment: An evaluation platform for general agents”, *J. Artif. Intell. Res.*, vol. 47, s. 253–279, 2013.

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

[6] Y. Combinator, ”Building Dota Bots That Beat Pros - OpenAI’s Greg Brockman, Szymon Sidor, and Sam Altman”, *Y Combinator*, 08-nov-2017. [Online]. Tillgänglig vid: https://blog.ycombinator.com/building-dota-bots-that-beat-pros-openais-greg-brockman-szymon-sidor-and-sam-altman/. [Åtkomstdatum: 18-mar-2018].

[7] I. Goodfellow, Y. Bengio, och A. Courville, *Deep Learning*. MIT Press, 2016.

[8] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, och W. Jaśkowski, ”ViZDoom: A Doom-based AI research platform for visual reinforcement learning”, i *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016, s. 1–8.

[9] N. F. McPhee, R. Poli, och W. B. Langdon, ”Field guide to genetic programming”, 2008.

[10] I. Millington, *Artificial Intelligence for Games*. Taylor & Francis, 2006.

[11] V. Mnih *m.fl.*, ”Human-level control through deep reinforcement learning”, *Nature*, vol. 518, nr 7540, s. 529–533, feb. 2015.

[12] OpenAI, ”Learning from Human Preferences”, *OpenAI Blog*, 13-juni-2017. [Online]. Tillgänglig vid: https://blog.openai.com/deep-reinforcement-learning-from-human-preferences/. [Åtkomstdatum: 26-feb-2018].

[13] D. Perez-Liebana, S. Samothrakis, J. Togelius, S. Lucas, och T. Schaul, ”General video game ai: Competition, challenges and opportunities”, *Proc. Conf. AAAI Artif. Intell.*, 2016.

[14] V. Rose, ”The International’s prize pool is, once again, the biggest in esports history”, *The Flying Courier*, 12-juli-2017. [Online]. Tillgänglig vid: https://www.theflyingcourier.com/2017/7/12/15959890/dota-2-ti7-prize-pool-biggest-prize-pool-esports-history. [Åtkomstdatum: 18-mar-2018].

[15] A. L. Samuel, ”Some Studies in Machine Learning Using the Game of Checkers”, *IBM J. Res. Dev.*, vol. 3, nr 3, s. 210–229, juli 1959.

[16] T. Schaul, ”A video game description language for model-based or interactive learning”, i *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, 2013, s. 1–8.

[17] B. Schwab, *AI Game Engine Programming*. Hingham, UNITED STATES: Charles River Media, 2004.

[18] D. Silver *m.fl.*, ”Mastering the game of Go with deep neural networks and tree search”, *Nature*, vol. 529, nr 7587, s. 484–489, jan. 2016.

[19] A. M. Turing, ”Computing machinery and intelligence”, *Mind,* vol. 59, nr 236, s. 433, 1950.

[20] O. Vinyals *m.fl.*, ”StarCraft II: A New Challenge for Reinforcement Learning”, *arXiv [cs.LG]*, 16-aug-2017.

[21] S.-C. Wang, ”Artificial Neural Network”, i *Interdisciplinary Computing in Java Programming*, S.-C. Wang, Red. Boston, MA: Springer US, 2003, s. 81–100.

[22] R. Yampolskiy och J. Fox, ”Safety Engineering for Artificial General Intelligence”, *Topoi*, vol. 32, nr 2, s. 217–226, okt. 2013.

[23] ”AlanTuring.net A Brief History of Computing”. [Online]. Tillgänglig vid: http://www.alanturing.net/turing_archive/pages/Reference%20Articles/BriefHistofComp.html. [Åtkomstdatum: 05-mar-2018].

[24] ”Learning through human feedback | DeepMind”, *DeepMind*. [Online]. Tillgänglig vid: https://deepmind.com/blog/learning-through-human-feedback/. [Åtkomstdatum: 26-feb-2018].