Primjena umjetne inteligencije i neuroevolucije u video igrama

Matkov, Dominik

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Matkov

PRIMJENA UMJETNE INTELIGENCIJE I NEUROEVOLUCIJE U VIDEO IGRAMA

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Matkov

Matični broj: 42779/14–R

Studij: Primjena informacijske tehnologije u poslovanju

PRIMJENA UMJETNE INTELIGENCIJE I NEUROEVOLUCIJE U VIDEO IGRAMA

ZAVRŠNI RAD

Mentor:

Dr. sc. Mladen Konecki

Varaždin, kolovoz 2019.

Marija Horvat

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Glavna i polazna teza ovog rada je umjetna inteligencija ili (Artificial Intelligence, kraće AI). Unutar rada povezivat ću umjetnu inteligenciju primarno uz video igre. Želio bih prikazati neki tok povijesti razvoja i napretka umjetne inteligencije. Kroz rad ću se kratko i doticati pojma machine learninga jer je on usko vezan uz AI i često se spominje u istom kontekstu. Prikazat ću vrste, načine primjenjivanja umjetne inteligencije i razlike između njih, a isto tako i algoritme koji se primjenjuju unutar svake vrste. Neka primarna metodika koje bih se dotaknuo je samoučenje, točnije primjena neuronskih mreža. Na to bi se nadovezivala moderna grana umjetne inteligencije, tj. primjena NEAT algoritma koju bih detaljno objasnio. Kao praktični dio rada napisana je videoigra tipa 2D platformera poput Super Maria i unutar nje imamo primjenu jednostavnog AI-a.

Ključne riječi: umjetna inteligencija; umjetne neuronske mreže; strojno učenje; neuron; aktivacijska funkcija; backpropagacija; NEAT; Bias

Sadržaj

1. Uvod

Danas se umjetna inteligencija koristi u mnogima granama znanosti i djelatnostima. Od zdravstva, mehanike, unutar proizvodnje, prometa, itd., a pogotovo u IT sektoru, gdje se dotiče i video igara. Unutar industrije video igara vrlo brzo se proširila uporaba umjetne inteligencije. Prilikom spominjanja umjetne inteligencije tu dolazimo i do pojma machine learninga koji se dotiče svega prethodno navedenog i koji se vrlo često interpretira kao sama umjetna inteligencija. Unutar ovog rada želio bih naglasiti da oni nisu isti i pokazati zašto je to tako.

Kao polazišna točka rada bio je video Setha Hendricksona ili bolje poznatog pod nazivom SethBling koji pojašnjava kako je u svojem radu napisao program MarI/O gdje je unutar već postojeće igre „Super Mario World“ implementirao proces neuroevolucije. Vidjevši to bio sam inspiriran da i sam napravimo nešto slično, ali unutar svoje igre. To je bio veoma ambiciozan pothvat i na kraju nisam uspio implementirati takvu opširnu neuronsku mrežu. Umjesto toga sam prešao na jednostavniju varijantu, a to je bilo pridodavanje umjetne inteligencije samim protivnicima kako se to nekad prije koristilo, uz pomoć A * search algoritma.

Svrha rada je prikazati kratku povijest umjetne inteligencije unutar video igara i razloge zašto se uopće koristi. Bez obzira na moj neuspjeh u prvobitnoj naumi ovaj rad će se doticati neuronskih mreža i neuronske evolucije, te će objasniti kako je nastala od prirodne evolucije. Pred kraj, rad će doticati kod iza praktičnog djela, tj. same video igre.

2. Metode i tehnike rada

Unutar rada koristio sam se primarno Unity Real-Time Developement platformom verzije 2018.3.9f1 Personal i to korištenjem 2D vizualizacije za cjelokupni prikaz 32-bitne video igre. Sve skripte su pisane unutar Microsoft Visual Studio 2017 IDE-a verzije 15.9.28307.518 korištenjem C# programskog jezika.

Koristio sam i library pod nazivom A* Pathfinding Project verzije 4.2.8. za implementaciju putanje te matrice koja je služila za prepoznavanje prepreka oko objekta. Pomoću tog library-ja napravio sam AI za neke protivnike unutar igre.

Za razvoj asseta prvo sam pronašao gotovi tileset sa OpenGameArt.Org koji se sastoji od 7 tilesetova, zatim sam svaki od njih obradio pomoću alata Gimp – GNU Image manipulation program verzije 2.10.8. Naravno kod obrade sam naišao na neke probleme poput ne konzistentnosti tilova kad bi se implementirali u unity kao asseti, uglavnom su bili različitih dimenzija i nisu se poklapale za gridom, kod nekih sam morao mijenjati veličinu, tj. rezoluciju, a i hitbox jer Unity ima unutar sebe alat koji automatski „prepoznaje“ obrub asseta i tako mu dodjeljuje hitbox što u mojem slučaju nije nikako odgovaralo veličini asseta. Nisam htio da to narušava user experience pa sam još uz sve to upotrebljavao i alate pod nazivom Tiled verzije 1.2.4 i Super Tiled2Unity verzije 1.5.1.

Za učenje i istraživanje ponašanja unutar neuronskih mreža koristio sam tečajeve na web repoziroriju Khan Academy-je gdje sam uglavnom koristio tečajeve za matrice, vektore i jedan tečaj koji se nazivao Games & Visualizations što mi je kasnije pomoglo u sveukupnom razumijevanju metodologije.

3. Razvoj umjetne inteligencije kroz povijest

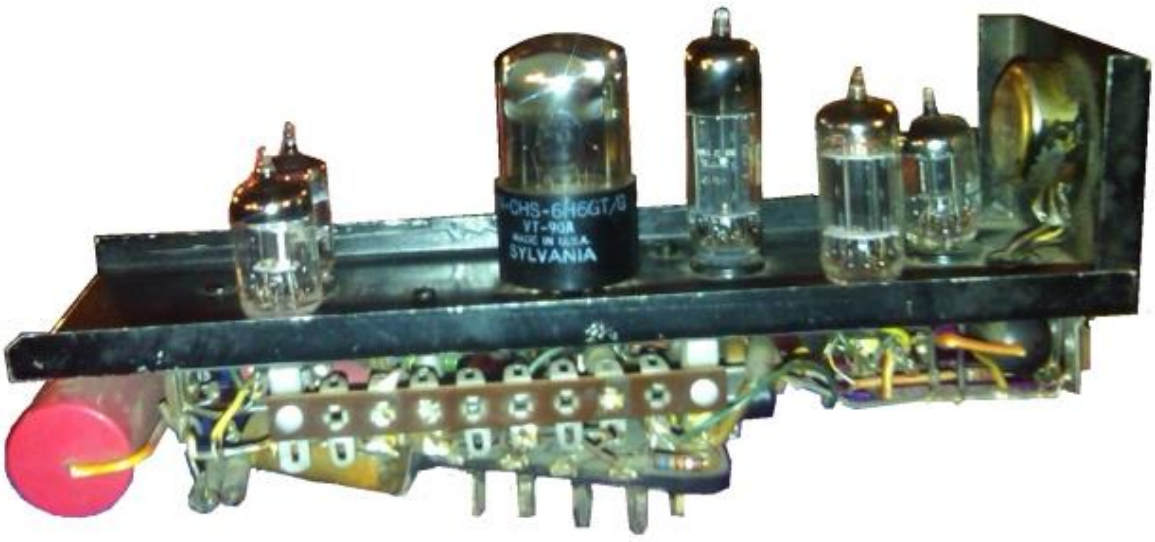
Još od davne prošlosti ljudi su sanjarili o tome kako će im sluge biti roboti, čak je i Homer pisao o mehaničkim tronošcima koji čekaju bogove oko stola. Od onda pa sve do danas kroz prošlost su se konstantno razvijale takve ideje, ali one su zaživjele tek u zadnjem stoljeću. Zajedno s napretkom tehnologije, došlo je i do napretka umjetne inteligencije. Prema Herbertu A. Simonu (1995) AI je dio računalne znanosti, ali je ujedno i dio psihologije i kognitivne znanosti. Prije razvoja tehnologije jedina inteligencija koja se mogla istraživati je bila inteligencija živih bića. Ovo poglavlje prikazuje sve napretke od začetaka pa do modernih dana razvoja umjetne inteligencije. Navodi svaki događaj koji je uvelike dao doprinos ili bio bitan preokret u istraživanju tog područja. Istraživanja polja umjetne inteligencije eksponencijalno rastu kako vrijeme prolazi.

3.1. Turingov test

Turingov test je nastao 1950. godine od strane znanstvenika Alana Turinga. Test se vrti oko jednog pitanja koje je Turing postavio, a to je „Mogu li strojevi razmišljati?“ (Turing, 1950, str 1). Test se sastoji od osobe koja postavlja pitanja i dva ispitanika od kojih je jedan ljudsko biće, a drugi stroj (računalo). Ispitivač mora odrediti tko je od njih što, s tim da računalo pokušava odgovoriti na pitanja onako kako bi odgovorio čovjek ili najbliže tome što može. Ako ispitivač više puta ne može odrediti razliku između njih to znači da računalo vjerojatno razmišlja. Osim testa u radu se spominje kako se računala vrlo vjerojatno može programirati da razmišljaju na inteligentan način. Ove iste godine Isaac Asimov je predložio tri zakona robotike. O njima se još i danas raspravlja s moralnog načela, ali su većinom odbačeni.

3.2. SNARC stroj

SNARC ili punim nazivom Stochastic neural analog reinforcement calculator označava početak neuroevolucije kao djela umjetne inteligencije. To je stroj koji je napravljen 1951. godine od strane Marvina Minsky i Deana Edmondsa, a simulira miša koji pronalazi sir unutar labirinta. Stroj se sastojao od 40 neurona, sinapsa koji su dodavali određenu težinu svakom neuronu ovisno o uspjehu u ostvarivanju određenog cilja. Od stroja je zabilježen izgled samo jednog neurona, koji je prikazan na slici 1. Cjelokupni stroj sam po sebi bio je rastavljen.



Slika 1. Neuron SNARC stroja (Image courtesy Gregory Loan, bez dat.)

3.3. Ferranti Mark 1 – upotreba AI-a u igrama

Ferranti Mark 1 je stroj napravljen 1951. na University of Manchester bio je prvi komercijalno računalo za sveopću upotrebu. Za njega su Christopher Strachey (dama) i Dietrich Prinz (šah) napisali programe koristeći umjetnu inteligenciju. To su bile prve inačice igara u kojima se pojavljuje umjetna inteligencija. Za razliku od njihovih veliki uspjeh je imao Arthur Samuel 1952. godine također s igrom dame u kojoj je AI 1955. kroz uspio steći takav napredak da je bio u stanju pobijediti amatera u igri dame usprkos svim limitima što se tiče hardvera i moći procesiranja. Od toga se nadalje uspjeh u stvaranju umjetne inteligencije mjeri pomoću video igara. Njegov rad je priznat na Darthmouth konferenciji 1956. na kojoj je i nastao pojam umjetne inteligencije.

3.4. ELIZA chat bot

Joseph Weizenbaum je napisao prvi chat bot, tj. prvi AI koji je razumio neke vrste prirodne komunikacije s čovjekom. Kao inpute je primao rečenice, koje je pomoću pravila za dekompoziciju rastavljao na ključne riječi. Output generira također koristeći se dekompozicijom, ali u to uvrštava i pravila za ponovno sastavljanje na temelju čega generira izlazne rečenice. Prema Weizenbaumu postoji pet problema s kojima se ELIZA suočava:

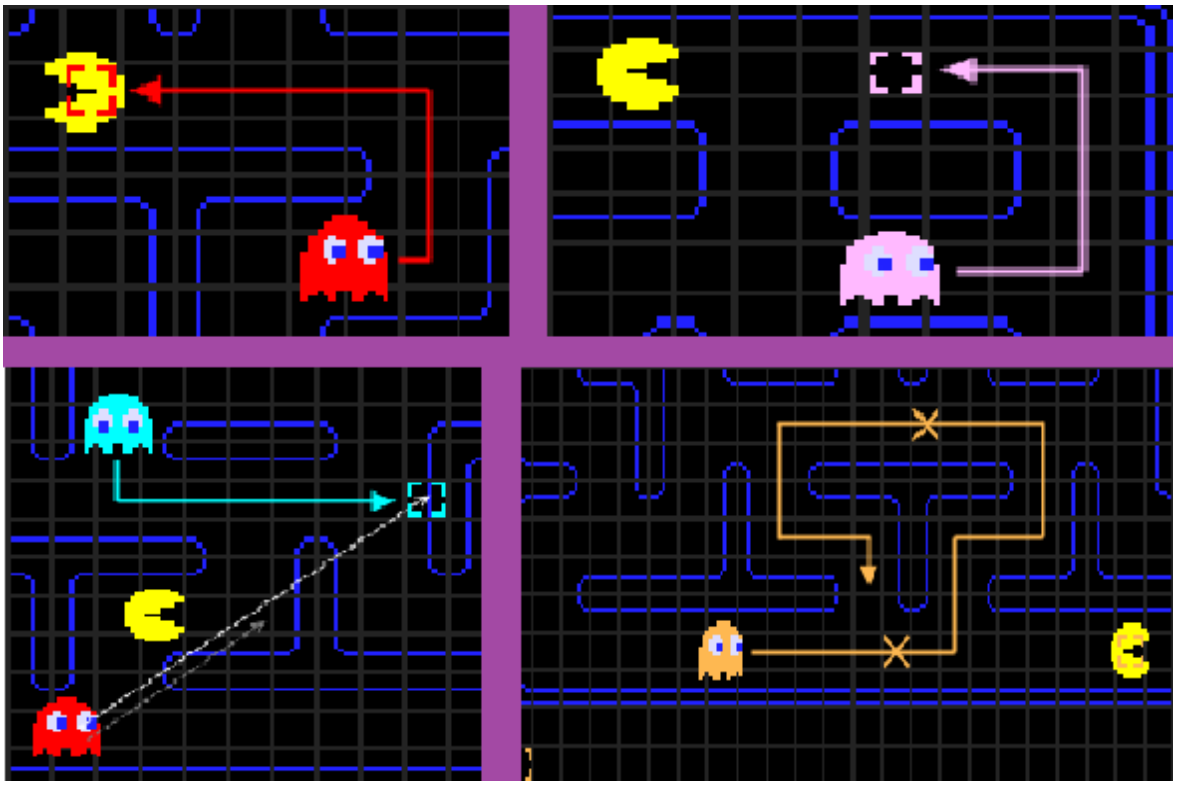
1. Identifikacija ključnih riječi
2. Otkrivanje minimalnog konteksta
3. Odabir odgovarajućih transformacija
4. Generiranje outputa kod nedostatka ključnih riječi

5. Ograničeni kapacitet za skripte

Kada dobije input rečenicu, ELIZA traži ključne riječi i dodjeljuje im određene vrijednosti po kojima ih tada rangira. Ponekad će zanemariti neke ključne riječi ako odredi da su njihove vrijednosti relativno manje u usporedbi s ostalima. Naravno dolazi do problema kad unutar inputa ne dobije ni jedan set ključnih riječi. Tada još nije bilo toliko memorije unutar računala da ono pamti prethodne instance inputa i da nauči na temelju njih.

3.5. Space Invaders i Pac-Man

Vjerojatno smo svi već upoznati s ovim naslovom nekad popularne igre. Ona je bila prekretnica za razvoj AI-a u video igrama, kada je izašla 1978 godine. Doduše to nije bio AI kakav znamo danas, ali većina tehnika koja se koristila tada još i sad koriste programeri polja machine learninga. Igra je imala mnoge prethodno ne viđene funkcionalnosti poput progresivne težine i različitih ponašanja protivnika. Kasnije kada je izašao „Pac-Man“ s njime je došao i algoritam traženja puta kojeg sam i koristio unutar praktičnog djela kako bih „dao život“ pojedinim protivnicima. Unutar Pac-Man-a bila su i četiri duha od kojih je svaki imao svoju personalnost.



Slika 2. Kretanje duhova u Pac-Man-u (Izvor: Birch C., 2010)

Crveni duh lovi igrača i postaje brži kada igrač pojede određeni broj hrane. Roza duh se pozicionira ispred igrača, plavi duh ima mješavinu ponašanja crvenog i roza duha iako ponekad ode lutati po labirintu. Narančasti duh je najmanje predvidljiv, njegova ponašanja su

programirana da budu nasumična. Ako se previše približi igraču on će ponekad otići u lijevi kut labirinta gdje je počeo.

3.6. Autonomno vozilo

Nakon ELIZE došlo je doba velikog napretka za polje umjetne inteligencije. Godine 1986. sveučilište Carnegie Mellon napravilo je prvo autonomno vozilo koristeći neuronske mreže u procesu. Prema video isječku iz nepoznate dokumentarne emisije (1986) nazvali su ga NavLab1. To je bio kombi u kojem su istraživači radili unutar njega prilikom testiranja. Mogli su mu prepisati preko njegove naredbe ukoliko bi došlo do veće greške. Kao inpute je koristio video kamere koje su snimale ispred njega. Npr. rubove ceste analiziralo je preko boje, teksture i intenzivnosti. Imalo je i laser koji je skenirao područje i udaljenost objekata u obližnjoj blizini. Pomoću tih inputa dolazilo je do primjerenih outputa, a oni su sadržavali pritisak kočnice, gasa i skretanje. Koristio je 3 procesa za rad, a to su bili: skeniranje okoline sa senzorima, mapiranje okoline i vožnja.

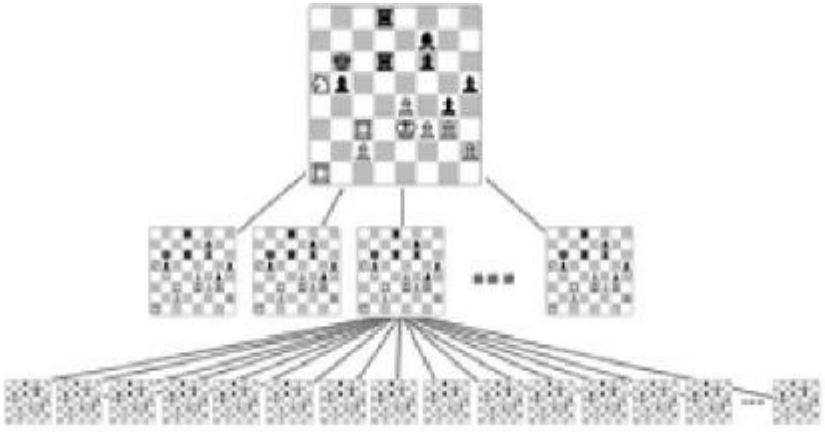
Prema Baluji S. (1996) evoluirani kontroleri (vozila) koristeći neuronske mreže proizvode veći uspjeh u dosad neviđenim situacijama nego što to proizvode modeli koji su trenirani uz pomoć algoritma backpropagacije koji je namijenjen isključivo za takve modele. Backpropagacija je metoda koja koristi Gradient descent (iterativni optimizacijski algoritam) za traženje minimuma funkcije kod težina dodijeljenih svakom neuroni o čemu možete više pročitati kasnije u radu. Tu dolazi do problema jer je tada metoda backpropagacije podložna tome da ostane u lokalnom minimumu dok su evolucijski mehanizmi bazirani na populaciji, tj. globalno i malo je vjerojatno da će podleći istom problemu.

3.7. Deep Blue vs Gerry Kasparov (šah)

Kako je umjetna inteligencija konstantno sve više i više napredovala, istraživači i znanstvenici su trebali sve veće prepreke, ali su ujedno morali i na atraktivan način prikazati rezultate njihovih istraživanja. Tako je 1997. IBM-ov Deep Blue pobijedio Gerrya Kasparova, grandmastera u šahu. Prema Andersonu M.R. (bez dat.) ta igra u nizu od 7 uzastopnih igri je trajala svega 19 poteza jer je računalo došlo do tako dobre pozicije da je Kasparov predao igru i čak optužio računalo za varanje. Rekao je da iza njega stoji neki grandmaster u šahu. Deep blue je igrao vrlo agresivno i nepredvidljivo, isto tako nije nasjelo na mamac koji mu je postavio Kasparov što ga je vrlo iznenadilo.

Deep Blue radi na principu primanja velikih količina podataka kao inputa. Model prikazan na slici 3 koji je koristio naziva se „Tree Search Problem“. Davali su mu poznate taktike otvaranja u šahu, a odabrane su one koje su priznate od većine grandmastera šaha. Kad bi

program došao do nekog problema, prepreke, tada je tim radio na tome da se ona otkloni, da se pronađe njen uzrok i način kako priječi oko budućih sličnih prepreka. Nakon svakog poteza računalo je pregledavalo bazu sa sličnim potezima i odgovorima na njih i tako je svaki puta došlo do odgovarajućeg rješenja problema.



Slika 3. Pojednostavljeni prikaz modela Tree Search problema (Izvor: Piech C., bez dat.)

Svaki unos u model, svaka „grana“ je stanje šahovske ploče i prikaz svih mogućih dozvoljenih poteza. Dalje se na svaki od njih primjenjuju algoritmi. Prije svega primjenjuje se evaluacijska funkcija. Pošto program ne može znati točno kako će igra izgledati u budućnosti i hoće li koji potez biti pobjeđujući ono uzima svako stanje u funkciju i daje joj realnu vrijednost. Npr. ako je sljedeće stanje poslije poteza ukloniti protivnikov pijun, a pritom neće izgubiti vlastiti tada ona određuje visoki broj za taj potez (stanje). Dalje je na to primijenjena funkcija koja minimizira maksimalni gubitak (loss), ali o tome više kasnije u radu.

3.8. Dolazak RTS igri

Kako je bilo mnogo napretka u primjeni AI-a, a to je omogućilo veće primjene u industriji video igara, došla je nova grana, a to su bile strategije. Jedna pod grana strateških igara jesu Real-Time strategy (skraćeno RTS) igre. To je vrsta igre gdje se istovremeno na potezu i igrač i protivnici, sve se dešava u realnom vremenu kao što naziv i govori. Prema Xu S. (bez dat.) prva u tom nizu koja je ukomponirala osnovni AI zvala se „Cytron Masters“, a ona glavna koja je postavila standarde za današnje RTS igre bila je „Dune II“ od izdavača Westwood Studios 1992 godine. Današnje igre poput StarCraft II za koje je također AI vrlo bitan faktor i koji ću spomenuti kasnije su naslijedile pravila i ponašanje iz Dune-a.
Neke metode koje su se koristile za AI uključuju A * search algoritam koji sam i ja koristio unutar praktičnog djela rada. Algoritam koristimo kada želimo doći od točke A do točke B na najbolji mogući način, tj. najkraćim putem. Princip rada algoritma je jednostavan:

- Pronađu polazišnu točku

- Pronađi odredišnu točku
- Definiraj put i putne točke (odredišta na putu)
- Kreći se dok sveukupni put ne postane manji od odredišne točke

Odredišna točka je mjesto gdje objekt (u kontekstu AI-a nazivamo ga „agent“) želi doći, put je cjelokupan put od trenutne lokacije objekta do lokacije gdje agent treba doći. Odredišta (točke) na putu se skraćeni dijelovi puta, svake n udaljenosti. Te točke algoritam koristi ako mu se mijenja odredište. U svakoj iteraciji algoritam provjerava na kojoj točki mora nastaviti put. Neće ići do kraja puta pa mijenjati putanju ukoliko joj je dinamično odredište i ono se tijekom putovanja mijenjalo, već će odrediti koja je točka trenutno najbolja za slijediti i na njoj generirati nastavak puta kako bi imao što manje odstupanje do novog odredišta. Ima nekih problema kod ovog algoritma, a to je da računalo mora veliku količinu podataka procesirati i nekad može kasniti s kretanjem, tj. s reakcijom.

Sljedeći problem koji nastaje je kada se kreće grupa objekata. Svi žele ići istim „najboljim putem“ i svi se drže toga, ali ne mare za ostale oko sebe. Tako nastaje linija objekata jednog iza drugog gdje se svi pokušavaju progurati naprijed umjesto grupe koja se kreće kao jedan. Kasnije je zbog tog problema Craig Reynolds razvio „Flocking“ algoritam. Su, Wang i Lin (2009) navode kako algoritam određuje virtualnog vođu kojeg prati cijela grupa agenata. Svaki agent je informiran gdje se vođa nalazi i dana mu je brzina kretanja prema objektu. Njihov cilj je održavanje te brzine kretanja. Pokazalo se da kada bi i malom broju agenata odredili njihovu brzinu, oni bi utjecali na kretanje cijele grupe.

3.9. OpenAI

Moderno doba umjetne inteligencije proslavio je OpenAI program. OpenAI je tvrtka osnovana od strane Elona Muska, a bavi se istraživanjem umjetne inteligencije. Ona je open source publikacija koja se nalazi na gitHubu javno dostupna (https://github.com/openai), a sastoji se od 83 repozitorija. Većinom je pisana u Pythonu. Veliku slavu program je stekao kroz igru „Dota 2“ kada je u rujnu 2017. na eSports sceni AI igrao protiv tada najboljih igrača svijeta navedene igre.

AI koristi koncept koji se naziva „Self-play“. Ono što on radi je da umjetna neuronska mreža (skraćeno ANN, Artificial Neural Network) za početka ima postavljene nasumične parametre vrijednosti neurona i tada mu daju da igra igru. Jedino kako utječu na njega je što mu daju pozitivne i negativne podražaje ovisno o tome što radi unutar igre. Kada prima štetu ili kada umre naravno proizvodi negativan podražaj, a kada on sam proizvodi štetu na protivnicima ili kada prouzroči završni udarac se nagrađuje pozitivnim podražajima. Znači koncept je vrlo sličan Freudovom konceptu u području razvojne psihologije o bihevioralnoj

kontroli. On govori kako se bilo koji problem može riješiti uz pomoć pozitivnih i negativnih poticaja, znači nagrađivanjem osobe za stvari koje smatramo da je učinila dobra, ali isto tako kažnjavanjem iste za stvari koje smatramo da nisu dobre. (Poulton, 1994, str. 137).

U početku će AI hodati bez cilja, raditi nasumične stvari, isprobavati. To je kao treniranje mozga djeteta, u početku ne zna što radi, isprobava stvari i tako mu se mozak razvija. Kada napravi dobru stvar i dobije pozitivan poticaj za to, on nastavlja raditi slične stvari, obrnuto za negativne stvari, pokušava ih izbjegavati, korigirati. S vremenom uči različite strategija, a isto tako se prilagođava na njih i pokušava doći do neke protu strategije. Kad je igrao protiv ljudi, većina njih je bila u čudu, vidjeli su neke nove strategije i zapravo su ljudi učili kasnije od OpenAI-a. AI je imao prednost što ga se ne može namamiti, ne može ga se zavarati, on gleda ono što je najbolje za njega u raznim scenarijima, a ne ono što bi možda moglo biti najbolje.

3.10. DeepMind – AlphaStar

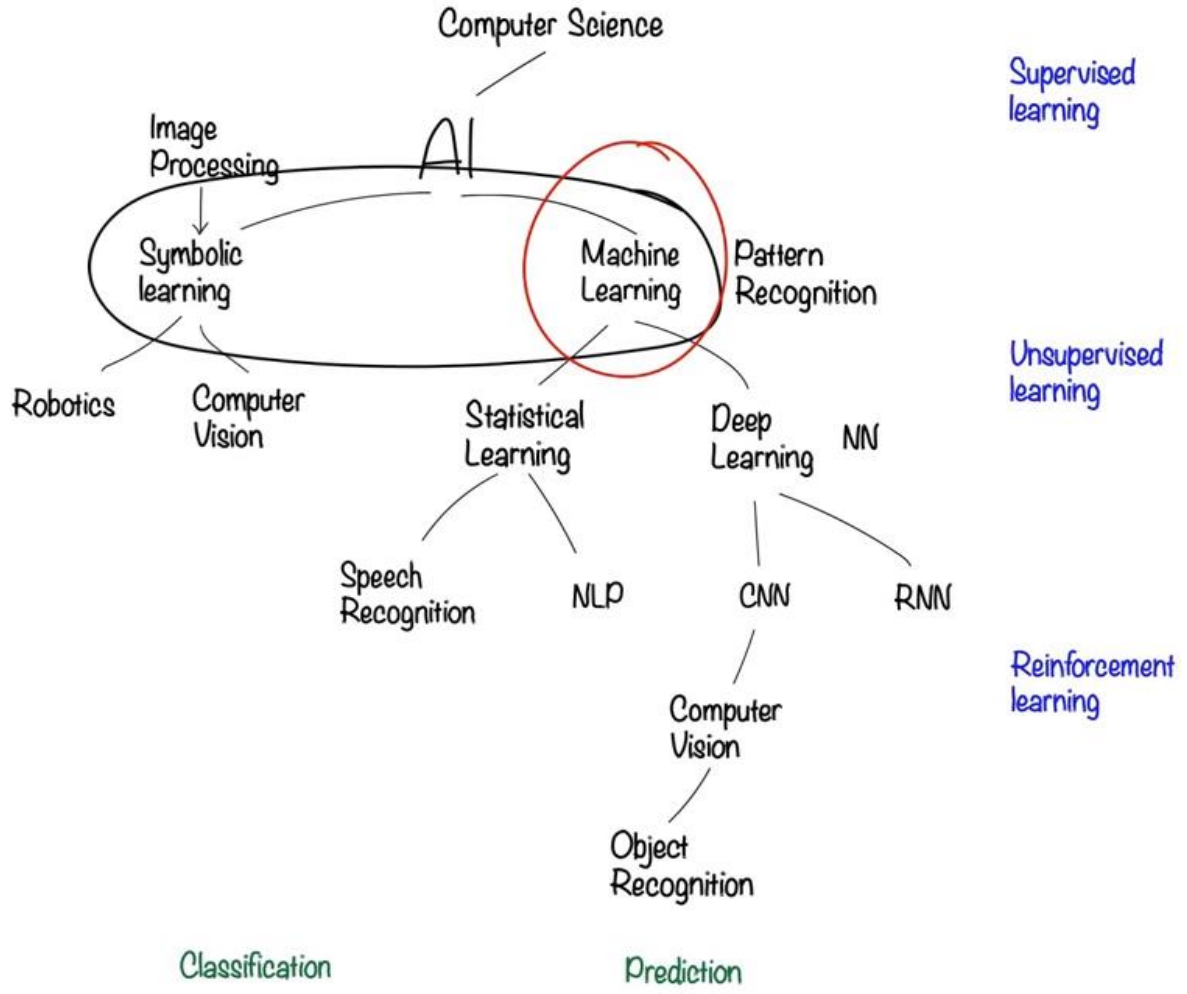
Nedavno, točnije 24. 01.2019. AI nazvan AlphaStar razvijen od DeepMind tima je prvi puta zaigrao protiv profesionalnog igrača StarCraft II scene. Igra je trajala sveukupno 7 minuta u kojoj je AlphaStar pobijedio profesionalnog igrača. Ovaj primjer je vrlo zanimljiv jer pokazuje koliko je zapravo napredovala umjetna inteligencija od igranja šaha i Pac-Mana do toga da pobjedi profesionalnog igrača RTS igre. Vrijednost mu dodaje činjenica koliko je zapravo kompleksno da AI igra na toj skali gdje su informacije koje dobiva kao inpute nesavršene, moraju se puno više obrađivati, svake sekunde je moguće napraviti približno 10^{26} različitih akcija.

Slika 4. Usporedba kompleksnosti operacija AI-a (Izvor: https://www.youtube.com/watch?v=DpRPfidTjDA)

AI je naučio samo jednu mapu (Catalyst) i jednu rasu (Protoss). Prije nego je pokušao igrati protiv profesionalnog igrača, AlphaStar se je trenirao gledajući reprize igri između ljudi, zatim igrajući protiv samog sebe, a zatim protiv zaposlenika DeepMind-a. Prema AlphaStar timu AlphaStar je naučio igrati igru koristeći duboku neuronsku mrežu koja je direktno trenirana uzimajući neobrađene podatke iz igre. Za obradu su korišteni koncepti učenja nazvani Supervised learning (nadzirano učenje) i Reinforcement learning (ojačano učenje). Oboje su koncepti koji spadaju pod pojam Machine learninga (skraćeno ML).

4. Klasifikacija umjetne inteligencije

Što je to zapravo umjetna inteligencija, što sve spada u područje umjetne inteligencije? Odgovore na ova pitanja ću pokušati objasniti u ovom poglavlju. AI se dijeli na više područja, od kojih možemo odvojiti Machine Learning. Prikaz podjele AI-a.



Slika 5. Podjela umjetne inteligencije

(Izvor: https://www.youtube.com/watch?v=2ePf9rue1Ao)

Podjela umjetne inteligencije prema Rameshu R. (2017). bazira se na akcijama i osjetilima koje posjeduje ljudsko biće. Jedna od njih je da ljudi mogu razgovarati međusobni i razumjeti se. To je grana nazvana prepoznavanje govora (Speech Recognition na slici). Isto tako ljudi mogu pisati i čitati, razumjeti znakove, to je polje koje se naziva prirodno procesiranje jezika (NLP na slici). Oba polja koriste statistiku kao glavnu granu za rad i postizanje rezultata i zato spadaju u višu granu koja se naziva statističko učenje (Statistical Learning na slici). Nadalje čovjek može vidjeti i vidom raspoznavati objekte, događaje, situacije oko sebe. To područje unutar AI-a nazivamo računalni vid (Computer Vision na

slici). Kako bi računalo dobilo ikakve informacije iz vida, mora moći analizirati podatke, a analizira ih u obliku slika. Ta radnja se naziva procesiranje slike (Image Processing na slici). Unutar simboličkog učenja spada i područje robotike. Kako bi se roboti uopće micali i znali što moraju raditi, trebaju im informacije iz okoline koje također obrađuju pomoću procesiranja slike. Zapravo je procesiranje slika nužan aspekt za rad simboličkog učenja i svih njegovih pod grana. Možemo primijetiti još jednu inačicu računalnog vida ispod područja dubokog učenja (Deep Learning na slici) koja služi kao medij za dohvaćanje podataka i slanje (CNN-u). Ljudi mogu razumjeti grupacije podataka u različitim bojama i oblicima, ekvivalentno tom nastalo je područje prepoznavanja uzoraka (Pattern Recognition na slici). Ono koristi strukturirane podatke i to velike količine podataka i dimenzije podataka i zato spada u područje nazvano strojno učenje (Machine Learning na slici). Kao jednu trenutno više zastupljeniju granu znanstvenici koriste neuronske mreže (ANN na slici). To je grana koja se zapravo razvila iz neurobiologije i proučavanja ljudskog mozga. Ona spada u kategoriju strojnog učenja jer radi na principu primanja velikih količina podataka i njihove obrade. Ako su to vrlo kompleksne neuronske mreže i koristimo ih za svladavanje kompleksnih problema tada ih uvrštavamo u kategoriju dubokog učenja (Deep Learning na slici). Nadalje postoje dvije vrste dubokog učenja, a to su konvolucijska neuronska mreža (CNN na slici) i ponavljajuća neuronska mreža (RNN na slici). CNN je mreža koja skenira ekran od lijeva na desno, odozgo prema dolje i koristi se za prepoznavanje objekata na nekoj sceni (slici). Za rad dobiva podatke koji su procesirani unutar računalnog vida i provođenjem tih podataka kao rezultat dobivamo prepoznavanje objekata na ekranu (Object Recognition na slici). I posljednja grana umjetne inteligencija koja se koristi neuronskim mrežama je prethodno navedena RNN. Ona pamti ograničenu prošlost i nju također koristi kao input u samu sebe.

U globalu dvije glavne podjele umjetne inteligencije su simboličko učenje bazirano na vidu i simbolima i strojno učenje bazirano na podacima. Tehnike koje se koriste u strojnom učenju služe nam za dvije stvari, a to su klasifikacija i predviđanje (ili regresija). Klasifikacija je stavljanje objekata u neki razred. Npr. je li osoba muško ili žensko Pokušavamo predvidjeti diskretni ishod. Predviđanje s druge strane je kada nema ponuđenog odgovora,već kao rezultat program određuje približnu prikladnu vrijednost. On predviđa rezultat. Kao primjer bilo bi određivanje starosti neke osobe. Pokušavamo predvidjeti rezultate u kontinuiranom ishodu. Postoji još jedan način razvrstavanja algoritma umjetne inteligencije u paradigme, a to su redom na slici:

- Nadzirano učenje (Supervised learning)
- Učenje bez nadzora (Unsupervised learning)
- Ojačano učenje (Reinforcement learning)

Algoritam spada pod nadzirano učenje ako je trenirano s podacima koji ujedno pružaju i odgovor. Njemu djelomično suprotno je učenje bez nadzora gdje je algoritam treniran s

podacima, ali nema eksplicitno naveden točan odgovor/e nego mreža sama prepoznaje uzorke i pomoću njih dolazi do rezultata. Ojačano učenje je vrlo slično učenju bez nadzora, ali nema označenih parova ulaza i izlaza u i iz algoritma. Ukratko algoritmu je predstavljen cilj koji on mora zadovoljiti kroz niz pokušaja i neuspjeha. Sve u svemu umjetna inteligencija je pojam koji obuhvaća sve ove koncepte i njime opisuje bilo kakvu vrstu primjene tih koncepata. Od pregršt-a različitih definicija, smatram da ovaj dijagram najbolje opisuje sljedeća: „AI, skraćenica za umjetnu inteligenciju, definira tehnologije danas u nastajanju koje mogu razumjeti, naučiti, a zatim djelovati na temelju te informacije.“ (Bothun, Lieberman, S. Rao, 2017, str. 2). Jedino što nedostaje definiciji je činjenica da umjetna inteligencija može evoluirati.

4.1. Simboličko učenje

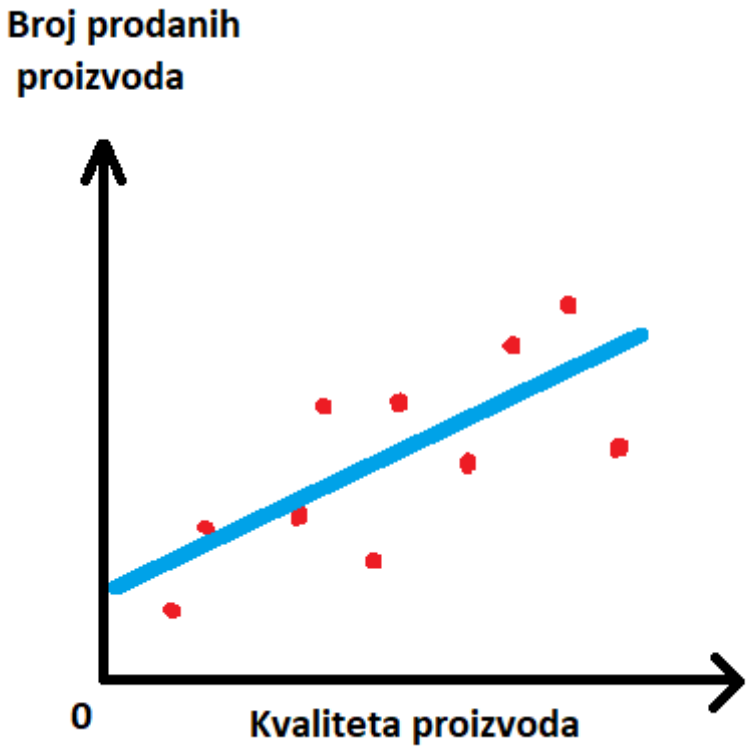
Kao što sam već spomenuo simboličko učenje je jedna od dvije glavne grane umjetne inteligencije zajedno s strojnim učenjem. Ono je trenutno zastarjelo ali se još uvijek koristi u nekim slučajevima, npr. kod automatizacije procesa u robotici. Kao osnovnu misao koristi ono što vidi kao reprezentaciju za probleme. Svi inputi kojima se koristi su simboli, slike. Namjena simboličkog učenja je da proizvede generalnu inteligenciju, onakvu kakvu bi zahtijevao od čovjeka. Strojevi koji su koristili taj princip su bili bazirani na tome da stvaraju output ovisno o inputu koji ima je dolazio u obliku simbola. To je super i radi kako treba kada su simboli u granici sigurnosti, kada dolazimo do situacija nesigurnosti, npr. prilikom predviđanja simboličko učenje ne može proizvesti takve outpute. I tu nam dolaze ANN-ovi. Ne želim ući previše opširno u ovu granu umjetne inteligencije jer ona nije osnovna misao ovog rada.

4.2. Strojno učenje

Strojno učenje je ono čime se umjetna inteligencija u globalu danas poistovjećuje, ali strojno učenje nije isto što i umjetna inteligencija. Moramo zapamtiti da je strojno učenje samo dio širokog spektra umjetne inteligencije, a ne ekvivalent tom pojmu. Zašto se onda miješa? Razlog tome je što se danas velika većina modernih procesa i problema rješava korištenjem ML-a ili nekog od njegovih grana i iz tog razloga je on sveprisutan i sve više se mijenja taj termin s terminom AI-a.

Prethodno poznavanju ML-a, sve se svodilo na to da su računala učila uz pomoć instrukcija koje su slijedila, a ljudi rade tako da uče na osnovi prethodnih iskustva. ML se bazira na statistici i teoriji vjerojatnosti. To su algoritmi i programi koji se unaprjeđuju kroz

vrijeme kada su podloženi novim setovima podataka. Tu već vidimo kako „napredak“ postaje jedan od ključnih pojmova unutar umjetne inteligencije.



Slika 6. Prikaz modela predviđanja ML-a (autorski rad)

U ovom primjeru modelu je predložen set podataka koji mu govori koliko je prodano jedinica proizvoda za određenu kvalitetu proizvoda. Prema kretanju cijena model određuje da se tu radi o linearnoj funkciji. Formula linearne funkcije za ovaj problem izgledala bi ovako:

$$y = mx + b$$

Gdje je y ono što želimo predvidjeti, u ovom slučaju broj prodanih proizvoda. Varijabla m predstavlja nagib linije, a x predstavlja vrijednost kvalitete proizvoda, dok nam je b vrijednost y osi. Kako bi ovo bilo primjenjivo u konceptu ML-a koristimo sljedeće oznake:

$$y = w_1 x_1 + b$$

Značenje se sada malo promijenilo:

- y označava output modela (još uvijek ono što želimo predvidjeti)
- w označava težine (weights) koje pridodajemo u svakoj dimenziji (1,2,3,...)

- x označava poznati input, ali također za svaku dimenziju, tj. set podataka
- b označava bias (os y) inače kontrolna vrijednost

Osim toga imamo i gubitak (loss) on nam označava udaljenost odstupanja rezultata od predviđanja funkcije. Čim je manji gubitak, bolji je model. Prema Pandeyu, P. (2018). postoji mnogo funkcija za računanje gubitka, jedna od njih naziva se L_2 gubitak ili kvadrirani gubitak. Izračunava se za jedan primjer treninga i to na sljedeći način :

$$(y - pretpostavka(x))^2$$

Istu funkciju nazivamo i funkcijom troška kada se radi na području cjelokupnog modela kao srednja vrijednost gubitka. Njezina formula glasi:

$$L_2 = \frac{1}{n}\sum_{(x,y)\in D}(y - pretpostavka(x))^2$$

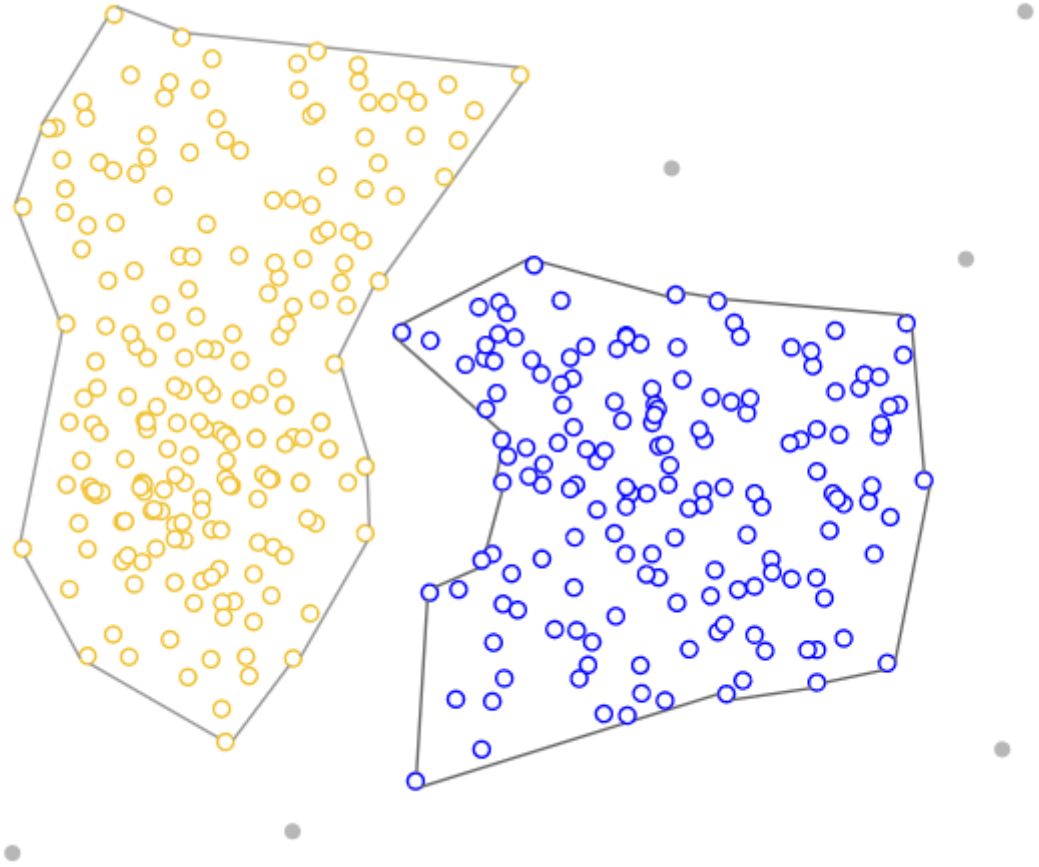
Gubitak svakog odstupanja je kvadrat vrijednosti odstupanja umanjena za vrijednost pretpostavke, što je uvijek pozitivno. Pomoću gubitka se trenira model sve dok ne dođe do najmanje vrijednosti gubitka. Problemi koji se mogu pronaći prilikom treninga je da model postane prezasićen od pretjeranog treniranja i on neće raditi na novim primjerima, tj. okolinama. Treba upamtiti da su glavna tri koraka za rad ML-a redom, izgraditi model, trenirati model i testirati model na novim setovima podataka.

4.2.1. Nadzirano učenje

Formalno ga nazivamo supervised learning. Kao što sam već prethodno napomenuo to je model strojnog učenja koji se koristi kada su nam poznati outputi modela. Kroz trening možemo s novim njemu nepoznatim inputima odrediti korektan output problema. Recimo da je broj prodanih proizvoda na prethodnom grafu uvijek bio 10, 15 ili 20, tada bismo taj model mogli svrstati unutar polja nadziranog učenja jer bi model tada znao točno očekivane outpute koji se od njega zahtijevaju. Unutar modela nadziranog učenja koristi se metoda spuštanja gradijentom. Cilj učenja je da trenira na podacima za koje zna output kako bi model mogao raditi kasnije na podacima za koje on ne zna output, ali zna kategorije u koje spada. Uglavnom se koristi za automatizirane procese i zadatke koji zahtijevaju inače puno vremena.

4.2.2. Učenje bez nadzora

Model istražuje skrivene uzorke od neoznačenih podataka. On dobiva inpute isto kao u kod modela nadziranog učenja, ali mu nisu dani očekivani outputi. Ono što model radi je otkrivanje skrivenih uzoraka unutar skupa podataka i strukture tih podataka i prema tome klasificira rezultate. Najpoznatiji algoritam koji koriste naziva se algoritam klasteriranja (grupiranja), eng. Clustering Algorithm. Prema Ester, Kriegel, Sander, Xu (1996). ima više vrsta algoritma grupiranja, ali najpoznatiji je „grupiranje na temelju gustoće“. Na slici 7 nalazi se primjer grupiranja podataka, tj. inputa prema gustoći što modelu služi za output kasnije.



Slika 7. Primjer grupiranja podataka prema gustoći
(Izvor: Google Developers, 2019).

Podaci se grupiraju u klastere na osnovu visoke gustoće identičnih podataka. Ukoliko ima podataka koji nisu blizu određenog klastera za njih model ne može sa sigurnošću odrediti gdje oni spadaju. Te podatke još nazivamo i područje buke.Ideja principa je da oko svake točke klastera u određenom radijusu mora biti neki broj točaka koji zadovoljava određeni prag kako bi se to moglo tumačiti kao dio zasićenja. Naravno kako bi se odredio rub klastera taj prag je smanjen i on označava „rubni dio“.

4.2.3. Ojačano učenje

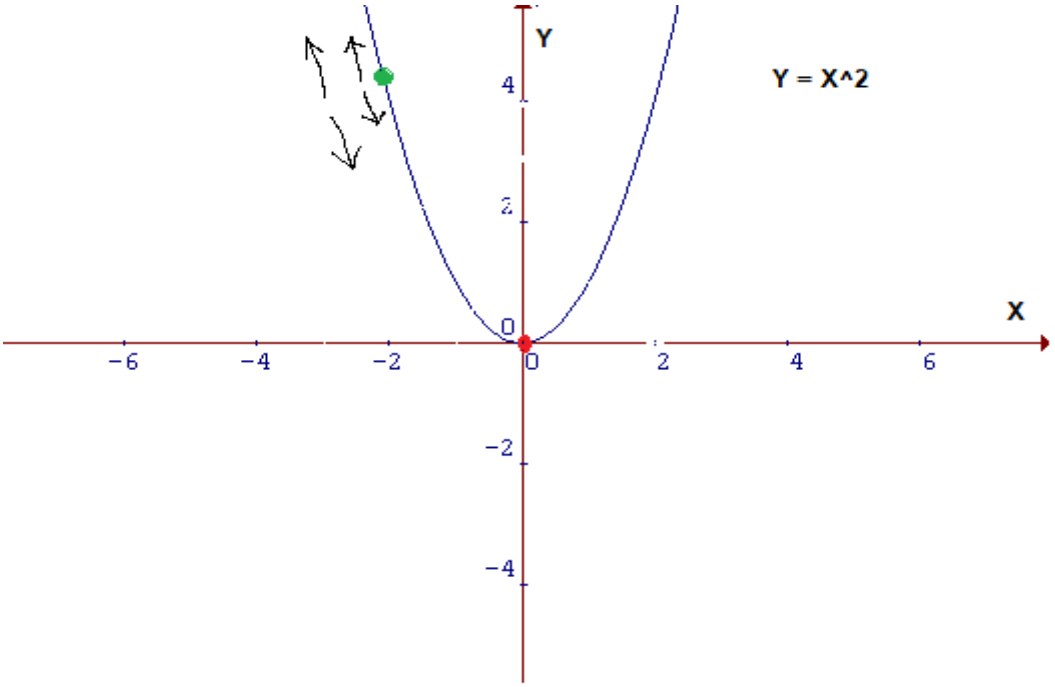
Ova vrsta učenja inspirirana je poljem bihevioralne psihologije. Agenti unutar modela kroz interakciju s okolinom uče kako optimizirati svoje ponašanje. Za razliku od prethodna dva modela, ojačano učenje (eng. Reinforced ili Reinforcement Learning) dobije output koji se od njega traži, ali nema nikakve inpute. Naravno kako će takav sustav uopće početi raditi? On radi na principu pokušaja i pogrešaka, ima nepredvidivo nasumično ponašanje, dok konačni ne dođe da nekakvih rezultata, tj. dok ne napravi nekakvu akciju. Tek tada kako bi model uopće mogao učiti, na njega se primjenjuje sustav nagrađivanja i kažnjavanja. Cilj modela je da dobije što više nagrada, a bude čim manje kažnjavan. Prema Karpathy A. (2016). u početku će vjerojatno agent konstantno biti kažnjavan, ali s vremenom će mu se posrećiti i taj nasumični odabir akcija će ga dovesti do pozitivne nagrade. Unutar modela se koriste gradijenti kako bi se dobivale akcije, ako je set gradijenata bio pozitivan i model je nagrađen, tada koristimo pozitivne vrijednosti kako bi povećali te određene gradijente koji su doveli do stanja uspjeha. Ukoliko je bio neuspješan s nekom kombinacijom gradijenta, model će se kazniti množeći njihovu vrijednost s „-1“. Takav pristup omogućuje da se smanji učestalost svih budućih akcija koje su bile loše u prošlosti, a poveća buduća učestalost svih akcija koje su donosile pozitivne rezultate u prošlosti.

4.3. Spuštanje gradijentom

Cilj svakog algoritma strojnog učenja je da smanji prethodno spomenutu funkciju troška. Čim je manja funkcija troška tim će s većom sigurnošću model određivati outpute novih ulaznih podataka. Funkcija troška zapravo može biti prezentirana u obliku parabole gdje je:

$$y = x^2$$

Kako bi minimizirali gubitak, moramo pronaći minimum funkcije, što je zapravo vrijednost varijable „x“. Unutar višedimenzionalnog sustava nije toliko lako pronaći minimum funkcije i iz tog razloga koristimo algoritam spuštanja gradijentom (Gradient Descent). To je jedan od najpopularnijih algoritama za optimizaciju modela, a upotrebljava se i u neuronskim mrežama.

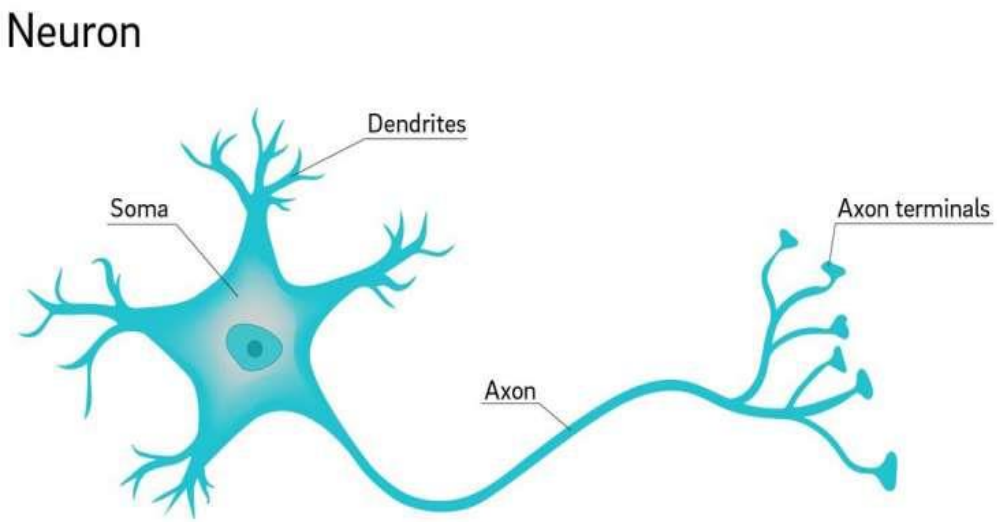


Slika 8. Prikaz spuštanja gradijentom, (Izvor: Pandey P. 2018).

Slika 8 grafički prikazuje spomenutu parabolu. Crvena točka je minimum funkcije, a zelena točka je mjesto gdje se nalazi model funkcije. Zelena točka ne može vidjeti gdje se nalazi crvena točka. Kako bi došla do tamo ona mora uzimati niz koraka prema gore i dolje. Pomoću derivacije funkcije model znade u kojem smjeru mora ići kako bi došao do minimuma. Grafički, deriviranjem model dobije tangentu na graf, koja mu pokazuje u kojem smjeru treba ići kako bi došao do minimuma funkcije. Isto tako ovisno i nagibu tangente može odrediti koliko veliki korak želi upotrebljavati. Taj korak se naziva stopa učenja. Cilj modela je da u što manje koraka dođe što je bliže minimumu funkcije. Ako uzme preveliku stopu učenja vrlo je vjerojatno da će preskočiti minimum. Isto tako ako model uzme premalu stopu učenja, vjerojatno će mu trebati ogromna količina koraka da se uopće približi minimumu što nije nikako efikasno. Jedan problem koji se javlja korištenjem ovog modela je što funkcija može završiti u lokalnom minimumu, a ne globalnom.

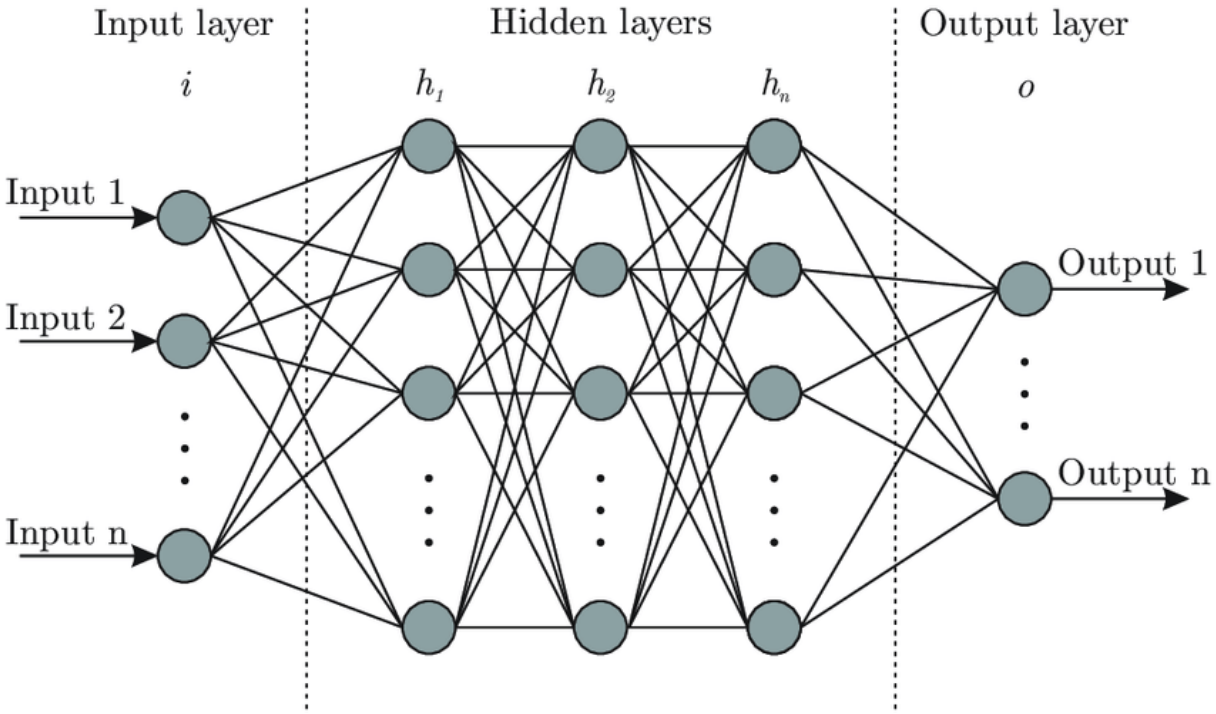
5. Princip rada neuronske mreže

Kako je tehnologija napredovala svijet je dobio mnogo brža računala i mnogo više podataka. S razvojem procesorske snage sve više se razvijala i grana neuronske inteligencije, tj. neuronskih mreža. Ona nasljeđuje koncept rada mozga životinja, a kasnije dolaskom dubokih neuronskih mreža nasljeđuje koncept rada ljudskog mozga. Neuronska mreža kao i mozak sastoji se od skupine neurona, oni su međusobno povezani sa svim ostalim skupinama neurona i svaki je u interakciji sa sljedećim. Slika 9 je prikaz neurona. On prima podražaje (s lijeve strane, na ulazne dendrite), procesira ih unutar aksoma i na temelju njih kaže tijelu što ono mora raditi kako bi reagiralo na te podražaje. Ima više vrsta neuronskih mreža koje ćemo obraditi kroz ovo poglavlje, isto tako ima i više načina na koji se one ostvaruju, raznim funkcijama, algoritmima.



Slika 9. Prikaz prirodnog neurona (Izvor: Baillot, 2018).

Unutar mozga imamo milijune ovakvih neurona, međusobno povezanih u jednu mrežu koja se naziva neuronska mreža. Isto tako se u umjetnoj inteligenciji koristi termin neuronskih mreža. Prikaz jedne mreže izgledao bi kao na slici 10.



Slika 10. Pojednostavljen prikaz neuronske mreže
(Izvor: Bre, Gimenez, Fachinotti 2017, str.4)

Na slici 10 prikazana je interpretacije neuronske mreže kakva je ona u stvarnosti unutar mozga živih bića. Kružići na slici su neuroni, oni su međusobno povezani tako da je svaki pojedini neuron prethodnog sloja povezan sa svakim neuronom u sljedećem sloju. Mreža se uvijek dijeli na tri vrste slojeva, ali sveukupan broj slojeva ovisi o kompleksnosti mreže. Ukoliko mreža ima sveukupno samo tri sloja tada je to jednostavna neuronska mreža. Ako mreža ima više od tri sloja tada takvu mrežu nazivamo dubokom neuronskom mrežom i to spada u područje dubokog učenja. Prvo imamo ulazni sloj (sloj „i“) koji predstavlja dendrite na prirodnom neuronu. Ovisno koliko želimo imati ulaza u mrežu, toliko imamo neurona postavljenih na ulazni sloj. Isto tako imamo i izlazni sloj (sloj „o“). On pak predstavlja aksiomske terminale unutar neurona živog bića. Iz tog sloja dobivamo outpute kao rješenja problema ovisno o inputima koje smo poslali u mrežu. Srednji sloj (sloj „h“), koji nazivamo i skriveni sloj, jer zapravo ne vidimo što se tamo dešava sastoji se od jednog ili više slojeva neurona. Na njega možemo gledati kao i na crnu kutiju. U srednjem sloju se primjenjuju algoritmi na funkcije i matematičke operacije kako bi dobili odgovarajući izlaz. Broj srednjih slojeva u principu trebao bi iznositi otprilike između broja ulaza i broja izlaza. Trebalo bi biti manje nego 2x ulaznih slojeva ili dvije trećine ulaznih + izlaznih slojeva kako bi se napravio adekvatan sustav (mreža), ali naravno ovo je pravilo kojeg se ne treba pridržavati, uvijek se mogu napraviti odstupanja.

Prije svega treba napomenuti kako svaki neuron (kružić) u sebi pohranjuje neku vrijednost između (ali ne uvijek) 0 i 1, osim ulaznog sloja. Naravno sve vrijednosti unutar mreže su brojčane vrijednosti. Na svakoj crti neuronske mreže, (slika 10) nalaze su vrijednosti koje nazivamo težinom (engleski weight, „w“). Težine su na početku nasumične, a model ih uči pomoću metode spuštanja gradijentom, tj. u kontekstu neuronshih mreža metodom backpropagacije. Iz svakog neurona ide jedna crta prema drugom što znači da svaki neuron ima jednu težinu prema drugom neuronu. Način na koji izračunavamo težinu svakog neurona je težinska suma inputa prethodnog sloja:

$$(w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n)$$

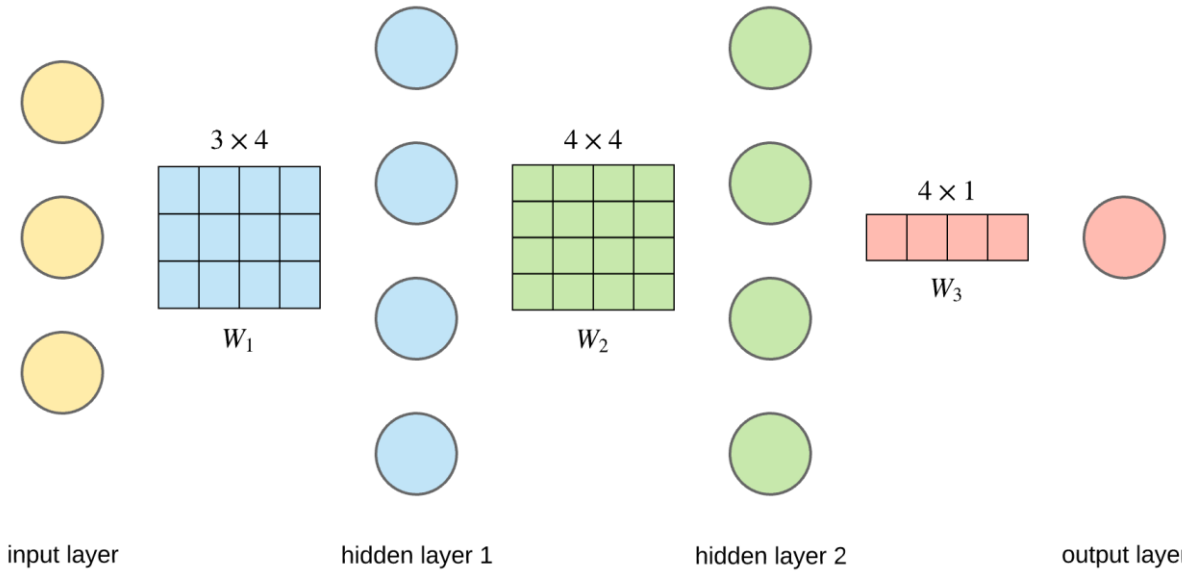
Gdje w označava težinu, x označava vrijednost inputa. Kada računamo s drugog sloja na treći, itd. x nam označava vrijednost prethodnog izračuna. Na taj rezultat se zatim primjenjuje aktivacijska funkcija o kojoj možete pročitati u sljedećem pod poglavlju. Jednadžba tada izgleda ovako:

$$g(w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n)$$

To još uvijek nije potpuna jednadžba jer svaki neuron ima svoju sklonost (eng. Bias) vrijednost, također o kojoj možete pročitati dalje u radu. S tim dobivamo potpunu jednadžbu izračuna vrijednosti svakog neurona unutar mreže, osim naravno ulaznih neurona.

$$g(w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n + b)$$

Ovaj sustav možemo zamisliti kao sustav množenja vektora i matrica.



Slika 11. Matematički prikaz neuronske mreže (Izvor: Dertat 2017)

Na slici 11 vidimo neuronsku mrežu s 4 sloja. Input (žuto), dva skrivena sloja (plavo i zeleno) i output (crveno). Kao ulaz uzimamo vektor vrijednosti inputa 1x3 i taj vektor množimo s matricom težina W_1 koja izgleda kao 3x4 matrica. Rezultat množenja tih komponenti ispada nam novi vektor 1x4 koji je reprezentacija vrijednosti prvog skrivenog sloja koji je prikazan plavom bojom. Nadalje taj vektor množimo sa sljedećim težinama u ovom slučaju W_2 matricom koja je dimenzija 4x4. Njihovim množenjem rezultira 1x4 vektor drugog skrivenog sloja reprezentiran zelenom bojom. I na posljetku taj novonastali vektor množimo s matricom težina prema output sloju koja je dimenzija 4x1 i dobivamo rezultat (Dertat A. 2017). Osim prethodno navedenog, za generiranje potpunog modela potrebna nam je i stopa učenja koja nam govori koliko će koja vrijednost koraka utjecati na težine. Isto tako nam je potreban i momentum koji govori koliko će prethodni rezultati utjecati na naše težine.

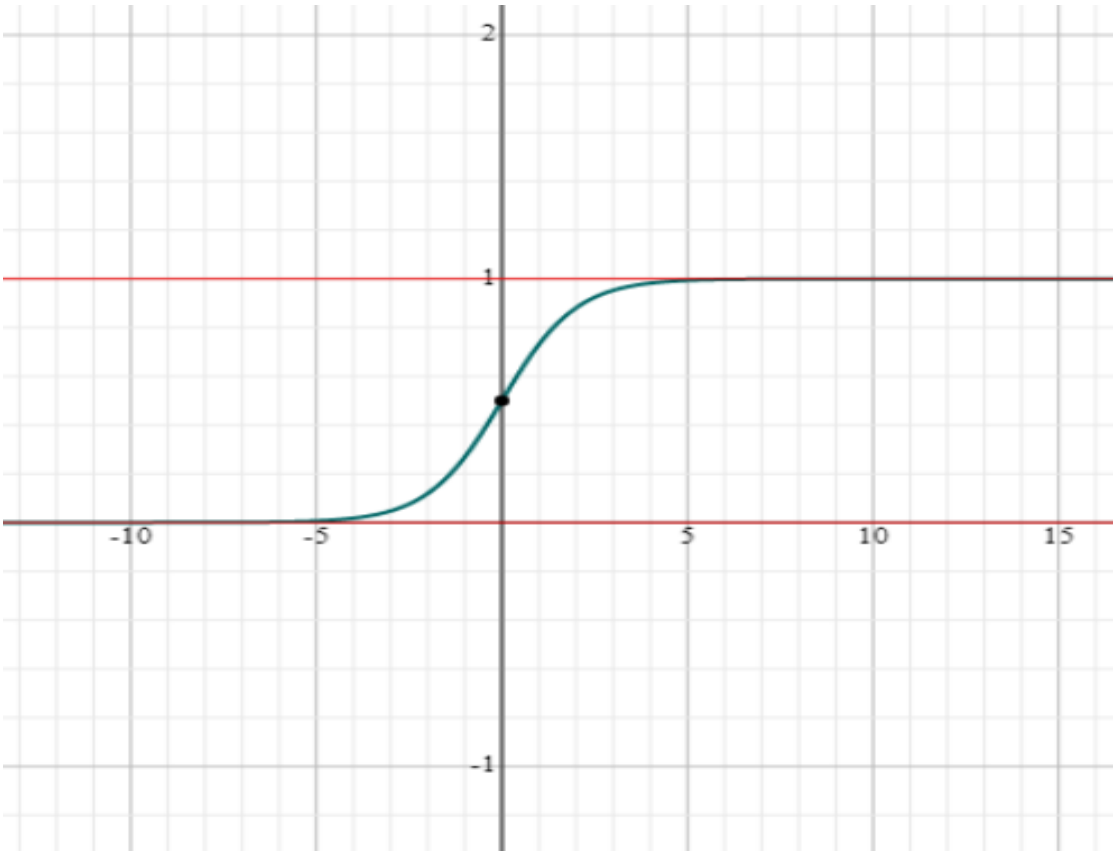
5.1. Aktivacijska funkcija

Da bi uopće neuronska mreže imala ikakvu funkcionalnost u njoj se koristi aktivacijska funkcija. Zapravo se ona koristi u svakom neuronu mreže. Njezina svrha je modificirati rezultat jednadžbe težinske sume inputa prethodnog sloja koju smo prvu spomenuli tako da ona uvijek vraća rezultate između nekog gornjeg i donjeg limita, inače je to 1 i 0, ali nije karakteristično za svaku aktivacijsku funkciju. Svrha aktivacijske funkcije je da definira vrijednost određenog neurona baziranog na inputima. Ukratko, govori nam hoće li se neuron aktivirati. Ako je rezultat funkcije blizu jedinici onda se smatra da je neuron aktivan, a ako je blizu nule onda se smatra neaktivan. Ovisno koliko je blizu jedinice tim je više aktivniji, a ujedno i značajniji. Značenje aktivnosti neurona nam govori, hoće li se on koristiti u izračunu za sljedeći sloj. Ako je neuron neaktivan, njegova vrijednost je nula. To možemo poistovjetiti s neuronima u mozgu, ako osjetimo neki ugodan miris, aktivirat će se pojedini neuroni, vjerojatno za sreću, glad, itd., a kad bi osjetili neugodan miris, aktivirali bi se neki potpuno drugi neuroni koji bi naznačavali naše krajnje outpute poput gađenja. U nastavku ću spomenuti od pregršt-a aktivacijskih funkcija samo dvije najznačajnije.

5.1.1. Sigmoid

Funkcija koja se kroz povijest najviše koristila za računanje vrijednosti neurona unutar neuronske mreže nazivala se Sigmoid funkcija. Ona pretvara rezultat u vrijednosti između 1 i 0. Njezina formula i graf:

$$S(x) = \frac{e^x}{e^x + 1}$$



Slika 12. Graf sigmoid funkcije (vlastita izrada)

- Za pozitivne vrijednosti funkcija pretvara input u vrijednosti blizu jedinice
- Za negativne vrijednosti funkcija pretvara input u vrijednost blizu nule
- Za vrijednosti blizu nule funkcija pretvara input u vrijednosti između jedinice i nule

Primjer jednog izračuna koristeći sigmoid funkciju:

$$g(w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n + b)$$

Za prikaz koristit ćemo nasumične vrijednosti i imat ćemo 2 inputa.

$$g(0.32 \cdot 4 + (-0.97) \cdot 6 + 1)$$

$$g(-4.54 + 1) => g(-3.54)$$

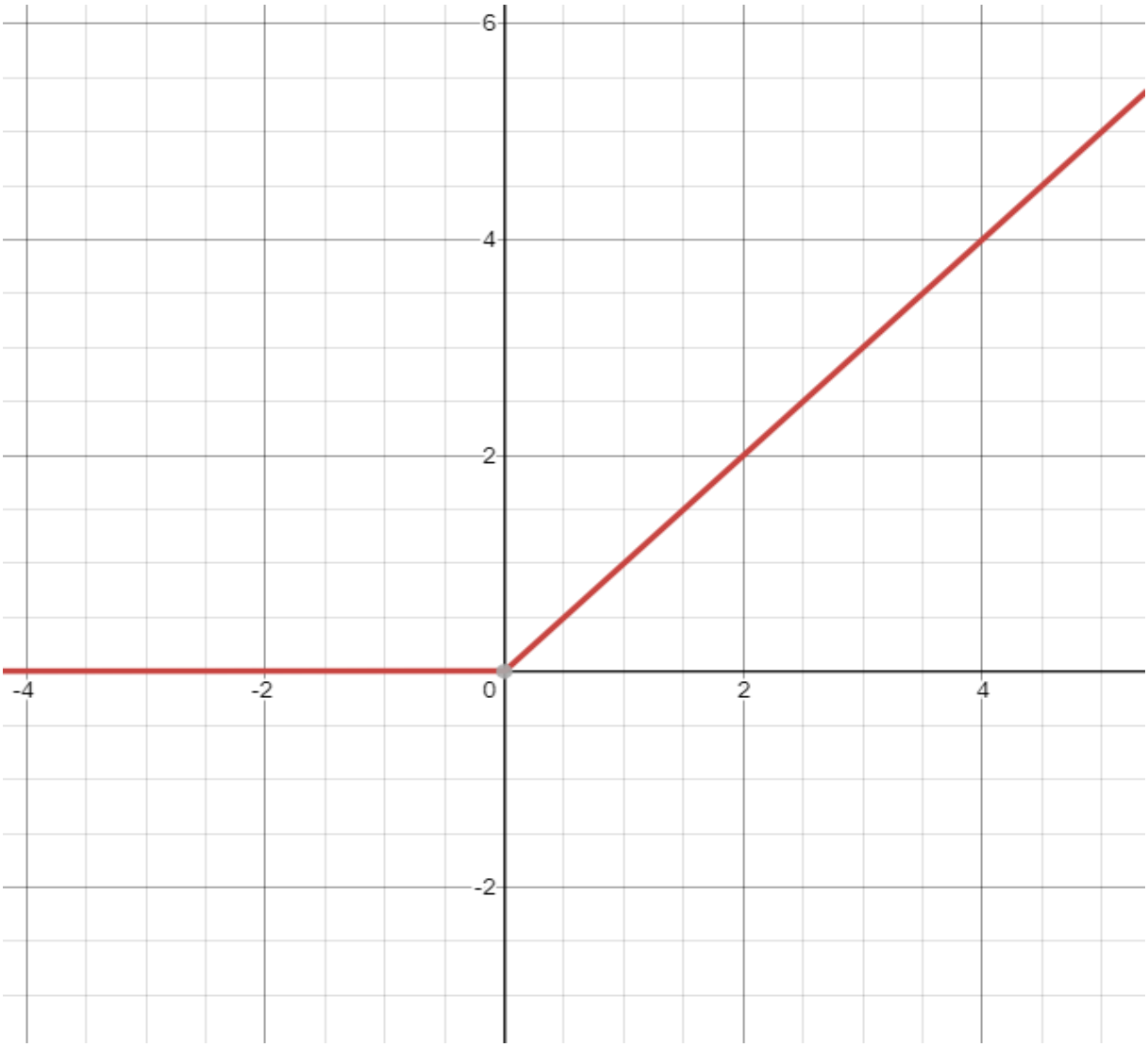
$$S(-3.54) = \frac{e^x}{e^x + 1} = \frac{e^{-3.54}}{e^{-3.54} + 1} = 0.028$$

5.1.2.ReLU

Sljedeća bitna funkcija naziva se ReLU (Rectified Linear Unit). Ona je danas najkorištenije i najzastupljenija aktivacijska funkcija. Koristi se gotovo u svim modernim

neuronskim mrežama dubokog učenja. Funkcija sama po sebi je mnogo lakša za procesirati i zahtjeva manje vremena obrade. Njezina formula i graf izgledaju ovako:

$$ReLu(x) = \max(0, x)$$



Slika 13. Graf ReLu funkcije (vlastita izrada)

ReLu je funkcija koja vraća maksimum između nule i parametra koji prima što znači da je vrijednost izračuna funkcije ili nula ili sam parametar funkcije. Funkcija nema negativne vrijednosti nego uvijek pokazuje vrijednost iz skupa pozitivnih brojeva.

- Za pozitivnu vrijednost X funkcija vraća tu vrijednost X
- Za negativnu vrijednost X funkcija vraća vrijednost nula

Znači ukoliko su negativne vrijednosti u pitanju, neuron nikad neće biti aktivan, a ako su pozitivne, neuron će uvijek biti aktivan, ali neće uvijek imati istu jačinu. Čim je neuron više

aktivniji tim je bitniji unutar ANN. Nadalje slijedi primjer izračuna koristeći ReLU funkciju s istim podacima:

$$g(-3.54)$$

$$ReLu(-3.54) = max(0, -3.54) = 0$$

Iz primjera vidimo da koristeći različite funkcije dobivamo različite rezultate. Bitno je da ovisno o problemu s kakvim se suočava model odaberemo pripadajuću aktivacijsku funkciju inače nam model neće raditi optimalno.

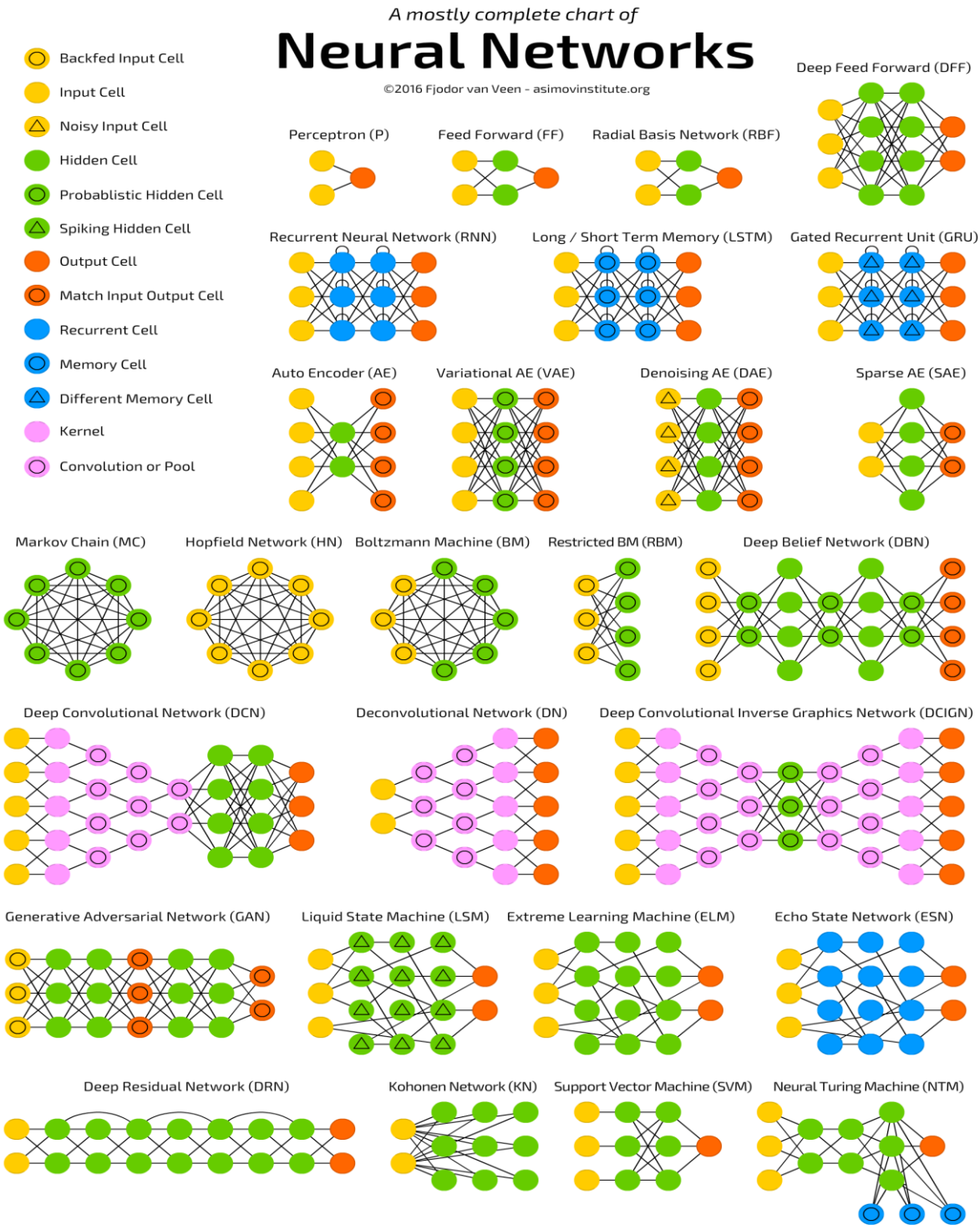
5.2. Sklonost ili Bias

Što je to zapravo bias? Svaki neuron u ANN ima svoj bias što znači da je cijela mreža sastavljena od više biaseva. Bias se može učiti isto kao što se mogu učiti i prilagođavati težine unutar neuronske mreže. Oni se također uče korištenjem metode backpropagacije. Bias možemo zamisliti poput praga, on odlučuje hoće li se ili neće neki neuron aktivirati i koliko će jako biti aktivan. On nam služi za značajno aktiviranje neurona. Biasevi služe za dodavanje fleksibilnosti modelu ANN-a za određene podatke.

Ako model odluči da pojedini neuron mora imati manji prag za aktivaciju, on to može napraviti, a radi tako da ovisno o tome koliko želi smanjiti aktivacijski prag, toliko proporcionalno povećava vrijednost biasa. Kao u primjeru izračuna funkcije ReLu, da nam je bias imao vrijednost 4, neuron bi bio aktivan jer bi mu tada prag aktivacije bio -4. Bias nam služi kao konsolidator. Isto tako ako model odluči da neki neuron treba biti aktiviran kad mu je prag recimo 3, tada on postavlja bias vrijednost za taj neuron na -3.

6. Podjela arhitektura neuronskih mreža

Ima više vrsta arhitektura neuronskih mreža koje ćemo obraditi kroz ovo poglavlje, isto tako ima i više načina na koji se one ostvaruju, raznim funkcijama, algoritmima. Prikaz svih trenutno zastupljenih umjetnih neuronskih mreža na slici 14.



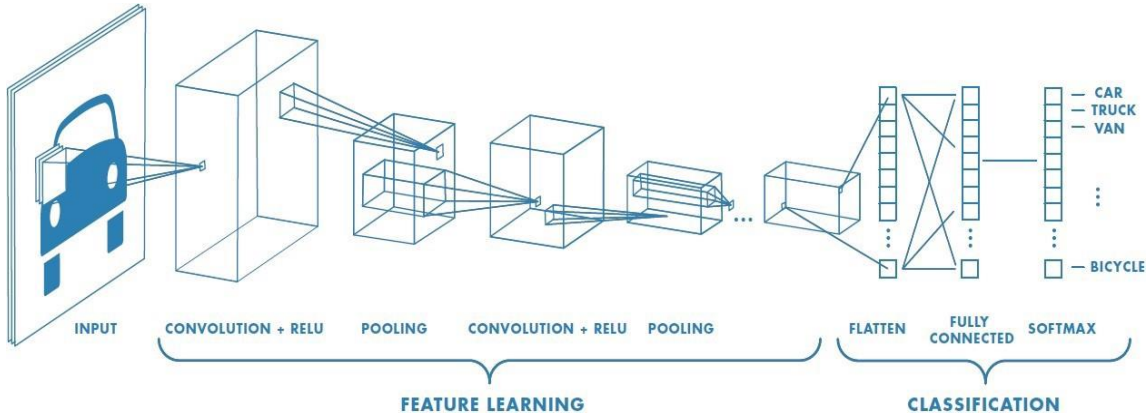
Slika 14. Slikoviti prikaz različitih arhitektura ANN-a (Veen, 2016).

Što se tiče ove podjele arhitektura ANN-ova želio bih samo proći kroz one najbitnije i najutjecajnije, a i meni najzanimljivije. Na ovaj dijagram možemo gledati kao na razne vrste mozga kod živih bića. Svaka arhitektura je dobra za neku drugu stvar. Recimo na primjer mozak šišmiša ima mogućnost korištenja ultrazvuka i on je za to „istreniran“, kad s druge strane ljudski mozak može obraditi puno više procesa, informacija i doći do kompleksnijih zaključaka. Svaki ima svoju ulogu i ne mogu biti zamijenjeni.

6.1. Konvolucijska neuronska mreža (CNN)

Kao što sam već i spomenuo u četvrtom poglavlju klasifikacije ANN-a, konvolucijske neuronske mreže su bitan dio tog područja. Konvolucijske neuronske mreže se primarno koriste kod prepoznavanja slika, tipa prepoznavanje lica na slici, prepoznavanje smješka što vjerojatno imate na mobilnom uređaju, isto tako se koristi i za captcha na internetu. Zapravo ono što mi prepoznajemo kako bi potvrdili da nismo „robot“ Google koristi kao inpute za treniranje neuronskih mreža.

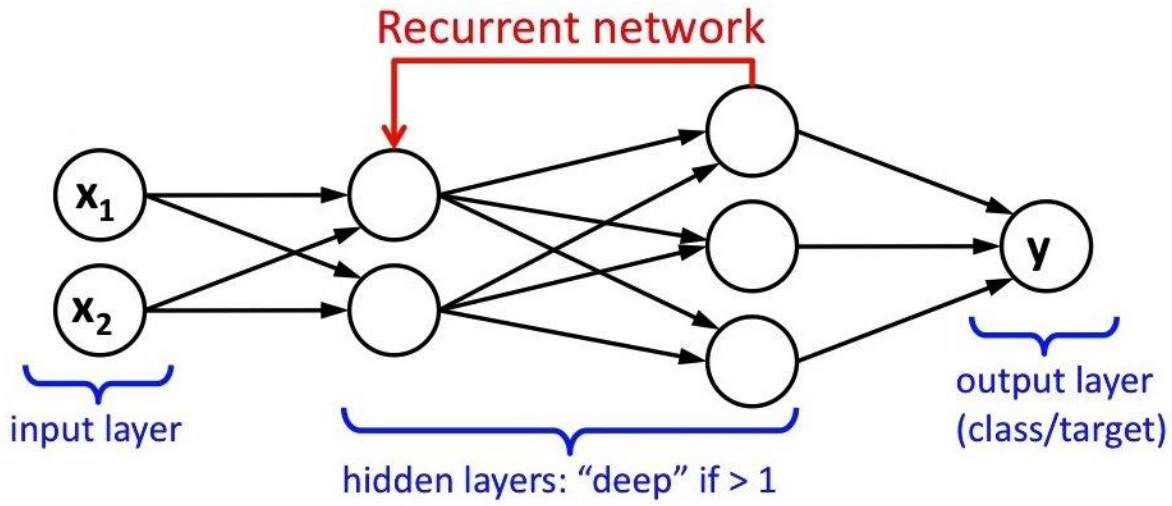
Lawrence et al.(1997.) napominju kako se konvolucijska mreža sastoji od skupa slojeva od kojih svaki sloj predstavlja jednu ravninu. Kao input mreža dobiva centrirane i normalizirane slike. Svaki sloj izvlači dio ravnine iz prethodnog sloja i analizira ga. Možemo to zamisliti kao da su pred nas stavili sliku na kojoj je masa ljudi na nekom koncertu i sad trebamo pronaći nekog poznanika unutar te slike. Prvo što naše oči vide je slika u potpunosti, dalje kako bi mogli bolje analizirati sliku, gledamo dio po dio slike. Zamislimo da smo sliku podijelili na 4 djela. Zatim recimo da smo pronašli dio gdje se nalazi masa ljudi na podiju, analizirat ćemo djelić po djelić tog podija, ako još uvijek nismo pronašli poznanika, analiziramo tribine, itd.. Na istom principu radi i konvolucijska neuronska mreža, prvo dobije cjelokupnu sliku kao input, svaki sloj neurona uzima neki dio slike od cjelokupnog inputa, sljedeći sloj dobiva manje dijelove dijelova, itd. ovisno o dubini mreže. Prikaz rada konvolucijske neuronske mreže možete vidjeti na slici 15.



Slika 15. Konvolucijska neuronska mreža (Saha, 2018)

6.2. Ponavljajuća neuronska mreža (RNN)

Ponavljajuće neuronske mreže se koriste uglavnom za audio i za podatke koji se mijenjaju kroz vrijeme, recimo govor. Još se može koristiti i za predviđanje stanja burze, generiranje teksta, prepoznavanje govora, generiranje nizova. Ove mreže imaju sposobnost kratkoročnog pamćenja na način da svaki neuron može umjesto slanja informacije naprijed, vratiti informaciju prethodnom neuronu. Znači osim što se računa unaprijed za svaki neuron, može se isto tako i vraćati na prethodni neuron za ponovnu evaluaciju, ali s korištenjem outputa od njemu sljedećeg neurona i to je vrlo korisno zbog stalnih promjena sa zvukom.

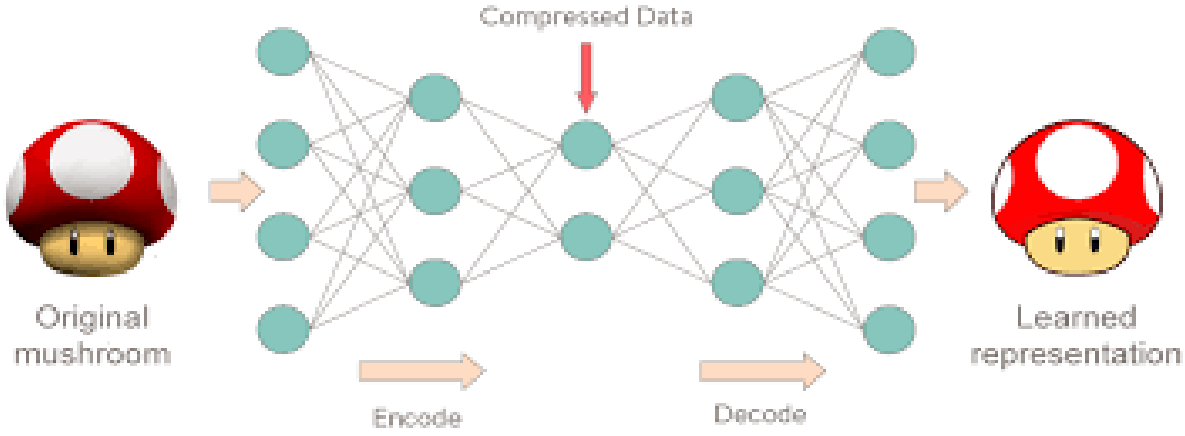


Slika 16. Prikaz ponavljajuće neuronske mreže (GitBook, 2017).

Ovu situaciju možemo zamisliti kao da radimo kućanske poslove na osnovu onoga što smo radili prošli dan. Ako smo danas npr. peglali sutra ćemo cijepati drva. Ako smo cijepali drva, sutra ćemo obrađivati vrt, a ako smo obrađivali vrt, sljedeći dan opet peglamo. S druge strane, ako je sunčani dan, radit ćemo poslove van kuće, a ako pada kiša radit ćemo poslove unutar kuće. Recimo da je prvi dan padala kiša i peglali smo (input nam je padanje kiše, a output je peglanje). Drugi dan je bilo sunčano i peglali smo prethodni dan pa smo danas cijepali drva (input nam je sunčano vrijeme, ali nam je input također onaj output od prethodnog dana [sustava], a to je da smo dan prije peglali). Sljedeći dan je opet padala kiša. Prema tome da smo jučer cijepali drva, trebali bi danas obrađivati vrt, ali pošto nam pada kiša odabrat ćemo ponovno peglanje za danas. Vidimo da mreža koristi inpute koji su joj dodijeljeni, ali ponekad mora koristiti i outputе prethodnih mreža ili neurona, ovisno hoće li krenuti padati kiša usred dana, a mi radimo neki posao vani. Nadam se da sam uspio objasniti princip rada ponavljajuće mreže.

6.3. Autoencoder

Novi pristup korištenju umjetnih neuronskih mreža i umjetne inteligencije predstavlja pojava autoencoder arhitekture. To je arhitektura koja izgleda ovako:



Slika 17. Prikaz mreže autoencodera (Dasgupta 2018).

Na prvi pogled malo čudno. Ono što radi mreža je da uzima inpute, uglavnom podatke koji su predstavljeni u obliku slika, „sažima“ ih dok ne dođe do uskog grla boce što je jedan ili dva neurona i onda ti neuroni pokušavaju predstaviti te sažete podatke nazad sljedećim neuronima u mreži pa oni sljedećim i to tako sve do outputa kako bi dobili sliku što sličniju originalnoj kakva je bila u inputu. Ovakva arhitektura neuronskih mreža uvijek ima isti broj neurona inputa i outputa. Iako ovo možda na prvu ruku izgleda beskorisno, ovakvi sustavi se koriste za generiranje novih podataka. Kad jednom mreža nauči koristiti recimo slova abecede, ona može pisati ta slova onako obrađena kakve je naučila.

Ovaj pristup je najlakše objasniti na primjeru učenja djece pisanju. Ona vide slova kako izgledaju, preko vida zamišljaju koje bi pokrete rukom trebala napraviti kako bi ponovno generirala takve simbole na papiru. Kad pokušaju pisati, slova neće ispasti točno onakvima kakve su ta djeca vidjela, ali će biti dovoljno blizu da budu razumljiva, da kad netko drugi pogleda, shvati koje je to slovo.

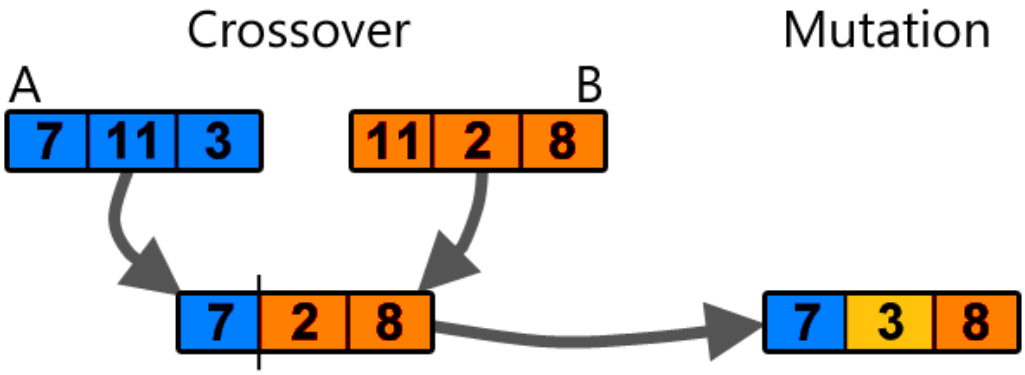
7. NEAT algoritam

Kada bi koristili puni naziv za ovaj algoritam on bi glasio Neural Evolution of Augmented Topologies ili na hrvatski neuronska evolucija proširenih topologija. Što to uopće znači? Da bi to objasnili trebamo krenuti od početka i to od biologije. Od teorije razvoja živih bića. U biologiji imamo genotip i fenotip. Genotip je genetička reprezentacija nekog živog bića dok je fenotip stvarna fizička reprezentacija živog bića. Zašto je to bitno? Bitno je jer NEAT koristi mutaciju kroz svoj model. NEAT možemo zamisliti kao prirodnu evoluciju živih bića. U ovom modelu evoluiramo/mutiramo mrežu mnogo puta tako da dodajemo (mutacijom), oduzimamo i prilagođavamo neurone na način koji proizvodi bolje rezultate.

Točno onako kako je generacija žirafa preživjela koje su imale duge vratove. One nisu oduvijek imale dugi vrat. S vremenom je došlo do mutacija unutar populacije žirafa, mutirane žirafe su mogle doseći do višeg lišća na drveću i one su mogle dulje preživjeti. Tu uvodimo fitness. To je funkcija koja određuje koliko je koja generacija sposobna. U 2D platformerima fitness bi odgovarao tome koliko daleko desno na ekranu dođe igrač i to u kolikom vremenu. Čim dalje dođe i u čim kraćem vremenu, tim je taj agent više fit. Funkcija je dobila ime po frazi „survival of the fittest“. Da nastavimo s pričom o žirafama, one žirafe koje su mogle pojesti više lišća bile su postigle veći fitness i s vremenom su mutirane generacije procesom reprodukcije preživjele, dok su generacije koje su imale mali fitness izumrle. Nisu mogli doseći lišće, a na njihovoj razini ga više nije bilo za jesti. Kroz vrijeme svaka generacija žirafa je mutirala i svaki put je opstala ona populacija koje je imala najveći fitness, znači ona koje je mogla doseći najviše lišće na drveću, tj. koja je imala najduži vrat. Kada bi to predočili u neuronsku mrežu. Možda je mogao postojati drugačiji put, možda je žirafa mogla biti više efikasnija da su bile brže i izdržljivije, a ne da su razvile duge noge i dugi vrat. Isto tako može postojati i bolje rješenje problema koje program nikad neće pronaći jer je odabrao drugi put. Evolucija nikada ne ide unazad, isto tako ni čovjek nije savršen koliko se to često spominje da je. Za vrijeme evolucije, mutacijom je čovjek naslijedio jedan živac koji u potpunosti krivo raste u našem tijelu. Više o tome na https://www.youtube.com/watch?v=cO1a1Ek-HD0. Isto tako i neuronske mreže mogu pogriješiti unutar evolucije/mutacije genoma i s tim će morati jednostavno nastaviti raditi.

Nakon ovog uvoda tehnički NEAT je evolucija koja koristi genetičke algoritme. Prije je bilo nezamislivo koristiti ovu vrstu umjetne inteligencije, ali sada kako se razvio pojam backpropagacije, to je bilo moguće za izvesti. Zamislimo mrežu ovako, svaka poveznica prema neuronu je jedna težina, na početku imamo 100 nasumičnih težina koje predstavljaju jednu generaciju. Nakon provođenja određenog zadanog

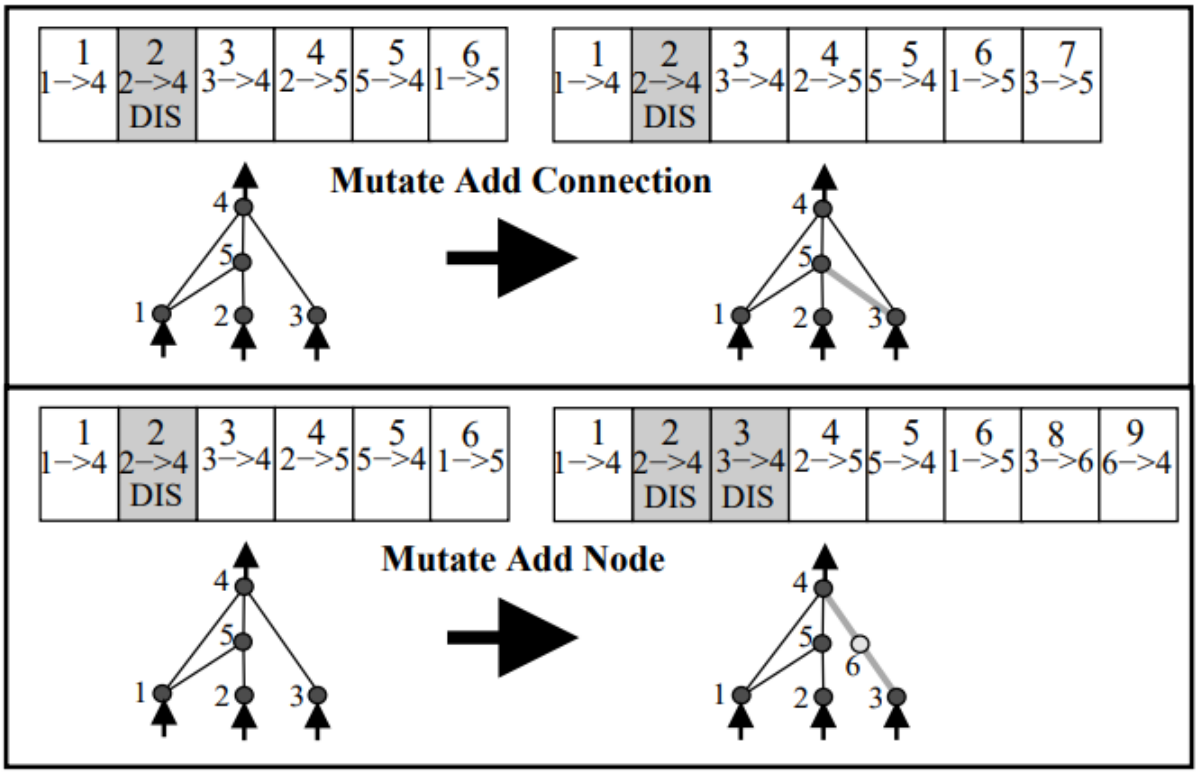
broja ponavljanja, gdje svaki agent nasumično isprobava doći do outputa pronalazimo kroz izračun fitness-a težine koje su recimo u najboljih 20% generacije. Te težine ostavljamo, ostalih 80 odbacujemo. Zatim radimo potomke kroz proces crossovera tako da nasumično uzimamo gene (genome) od roditelja i od njih nastaju potomci. To se radi tako da se svaka težina roditelja pretvara u jednodimenzionalan niz.



Slika 18. Crossover i mutacija genoma (Abrandao, 2015).

Na prikazu slike 18 možemo vidjeti proces stvaranja novog potomka i njegovu nasumičnu mutaciju. Naravno mutiranje se ne primjenjuje na svakog potomka. Nakon mutiranja, vrijednost se nazad pretvara u decimalni broj i zajedno sa ostalih 99 se prosljeđuje nazad u neuronsku mrežu i tako nastaje druga generacija. Kroz trening druge generacije, opet se uzimaju oni genomi koji su ostvarili najbolji fitness i cijeli proces se ponavlja. Fenotip je fizički kostur neuronske mreže poput onoga sa slike 10.

Postoje dva tipa mutacije koji mogu nastati. Prikazano na slici 19. možemo protumačiti da su brojevi 1,2 i 3 ulazi u neuronsku mrežu, broj 5 je skriveni sloj, dok je broj 4 output neuronske mreže. prvi tip mutacije je dodavanjem dodatne veze između genoma (gornji dio slike) gdje nastaje nova poveznica između ulaznog neurona 3 i neurona skrivenog sloja 5. Na donjem dijelu slike možemo vidjeti mutaciju dodavanjem novog neurona 6 u mrežu. On nastaje između veze neurona 3 prema outputu 4, tim procesom nastaju i dvije nove veze, ona između neurona 3 i 6, te ona između neurona 6 i 4.



Slika 19. Strukturalna mutacija genoma (Stanley, Miikkulainen, 2002).

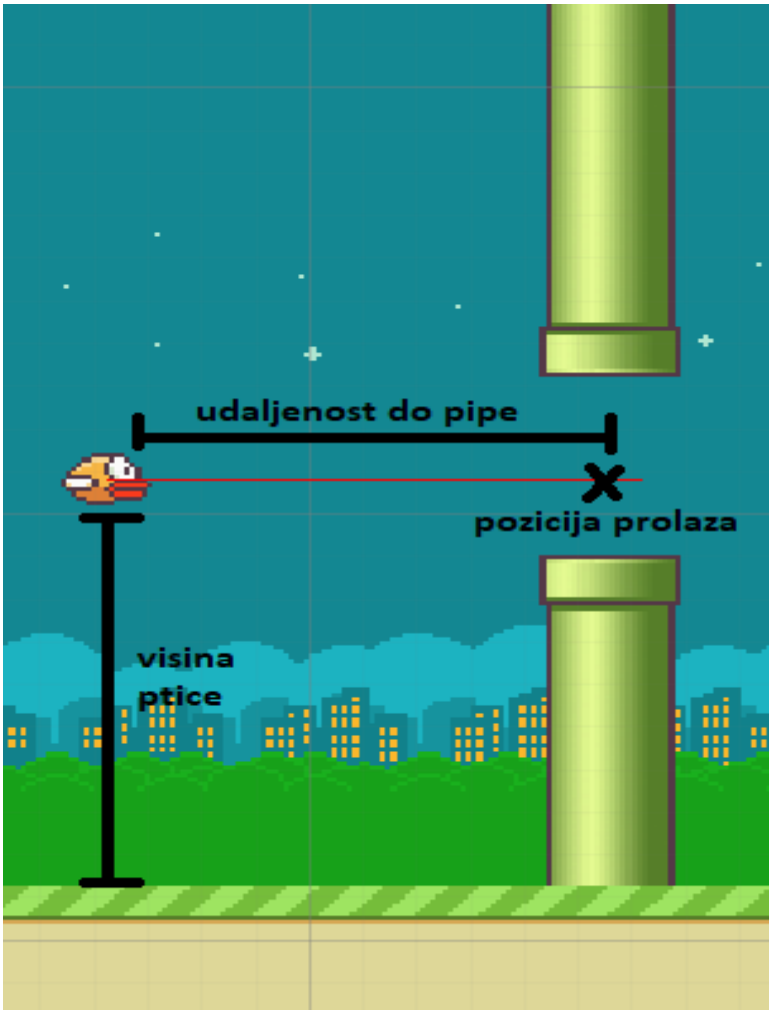
Kao i u stvarnom životu znamo tko su nam predci tako i genomi unutar neuronske mreže moraju znati tko su njihovi. Prema Stanley, Mukkulainen (2002). prilikom svakog nastajanja novog gena, inkrementira se globalni broj reprodukcije (inovacije) i dodjeljuje se tom novom genomu. Time dobivamo kronološki poredak nastajanja genoma populacije. Svaki put kad se dešava reprodukcija, novonastale generacije genoma dobivaju inovacijski broj od roditelja. To omogućava sparivanje gena koji odgovaraju jedni drugima, a oni koji ne odgovaraju, potomak nasljeđuje od roditelja s većom fitness vrijednosti. Posljednji bitan korak kod evolucije je razdvajanje određenih vrsta od globalne populacije, kako bi imale vremena prilagoditi se među sobom. To je problem koji je također riješen pomoću broja inovacije. Kako bi pronašli genome iste vrste moramo uspoređivati genome međusobni i pronaći one koji imaju najmanji broj nespojivih gena i viška gena. Čim više nespojivih gena postoji između para genoma, tim manje evolucijske povijesti oni dijele među sobom. Funkcija za izračun bliskosti genoma glasila bi:

$$\delta = \frac{C_1 E}{N} + \frac{C_2 D}{N} + C_3 \cdot avg(W)$$

Gdje su C_1, C_2 *i* C_3 faktori koje koristimo za normalizaciju, a N koristimo za normalizaciju veličine genoma (može biti postavljen na 1, ako se genom sastoji od 20 ili manje gena). E predstavlja broj viška gena, dok D predstavlja broj nespojivih gena. W koristimo kao srednju vrijednost razlike između težina spojivih gena.

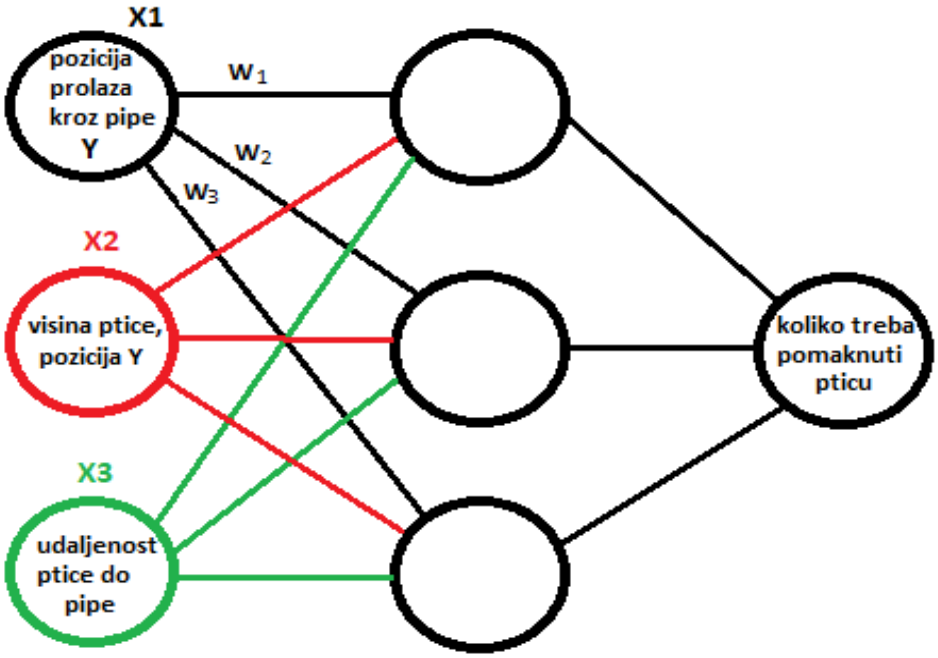
8. Praktični rad

Kao dio ovog rada napravio sam neuronsku mrežu baziranu na paradigmi nadziranog učenja korištenjem metode backpropagacije, a kao aktivacijsku funkciju koristio sam TanH. Igra koju moja neuronska mreža treba naučiti bazirana je na igri „Flappy bird“, ali uz neke izmjene.



Slika 20. Prikaz igre na kojoj je treniran agent (vlastita izrada)

Ptica (u ovom slučaju agent neuronske mreže) mora proći između pipa, ako se zaleti u njih nastavlja dalje. Pipe nemaju rigidbody, nego samo collider na sebi, tako da su zapravo transparentne za pticu. Ispred sebe ptica ima raycast tako da može detektirati pipe i događaj kada se sudari s njima. Na slici 21 je prikaz izgleda neuronske mreže projekta.



Slika 21. Prikaz neuronske mreže praktičnog rada (vlastita izrada)

Mreža se sastoji od tri sloja. Ulaznog sloja, jednog skrivenog sloja i izlaznog sloja. Ulazni sloj sastoji se od 3 inputa, skriveni sloj sastoji se od 3 neurona, a izlazni sloj proizvodi samo jedan izlaz. Kao inpute u mrežu odabrao sam: **1. poziciju prolaza kroz pipe**, to možemo referencirati kao pozicija Y objekta pipe pošto su obje pipe stavljene u jedan objekt kao djeca tog objekta. Pozicija na kojoj se pipe nalaze je ujedno i pozicija prolaza kroz pipe. To je objekt koji ne instanciramo već se on pomiče kada mu pozicija X bude određeni negativan broj kojim on dospijeva izvan slike. Objekt se pomiče na desnu stranu slike kao novi ulaz sa različitim nasumičnim vrijednostima Y osi tako da se pozicija konstantno mijenja. Sljedeći ulaz je **2. visina ptice** (agenta) koju referenciramo kao pozicija Y osi ptice u svakom okviru (eng. frame) igre. Zapravo je to pozicija na kojoj se nalazi ptica, jer joj je X os konstantna i po njoj se ne može micati, isto kao ni po Z osi koju u potpunosti zanemarujemo jer je ovo 2D projekt. Posljednji ulaz **3. udaljenost ptice od pipe** računamo kao poziciju X osi objekta pipe umanjenu za poziciju X osi objekta ptice.

Te inpute spremamo u listu inputa i to radimo za svaki neuron. Isto tako imamo listu outputa koja služi kao input sljedećem neuronu. Na slici inputi su označeni s X_1, X_2 i X_3. Svaki input množi se sa težinama koje određuje sama neuronska mreža i koje su nam i najvažnije za mrežu. Težine su označene sa W_1, W_2 i W_3. Kroz cijeli proces toka ove neuronske mreže ona izračunava i prilagođava težine. Isto tako mreža ima i bias vrijednosti koje dodaje, odnosno oduzima svakom umnošku težine i inputa prethodnog neurona. To radi za svaki neuron skrivenog (srednjeg sloja) neurona. Kada bi smo na kraju treninga (kada

smo zadovoljni s rezultatima koje agent prikazuje) spremili sve vrijednosti težina tada bi to bio spremljeni prikaz istrenirane neuronske mreže. Te težine u teoriji mogli bi smo primijeniti na bilo koju sličnu situaciju u igri i agent bi znao to prijeći. Posljednji sloj koji se nalazi u mreži je sloj izlaza gdje kao izlaz agent dobiva vrijednost brzine kojom se mora pomaknuti da bi prošao kroz otvor između pipa.

Ova neuronska mreža prikaz je nadziranog učenja jer imamo očekivani output. Očekivani output. Na temelju očekivanog outputa računamo grešku koja nam služi za proces backpropagacije. Grešku ponovno prosljeđujemo u neuronsku mrežu.

```
if (hit.collider != null)
{
    float odstupanje = pipa.transform.position.y - ptica.transform.position.y;
    output = Run(pipa.transform.position.y, ptica.transform.position.y,
            (pipa.transform.position.x - ptica.transform.position.x), odstupanje, true);

    velocityPtice = (float)output[0];
}
else {
    velocityPtice = 0;
}
```

Slika 22. Prikaz koda određivanja greške (vlastita izrada)

Na slici 22 možemo vidjeti baš to navedeno određivanje greške. Varijabla „hit“ je zapravo raycast (crvena zraka na slici 20) kojom određujemo kada se ptica zaleti u objekt pipa. Kada se to desi, gledamo gdje se točno ptica zaletjela i računamo udaljenost od tog mjesta do očekivane pozicije gdje je trebala biti na osi Y kako bi prošla kroz sredinu pipa. Varijabla „velocityPtice“ označava nam brzinu kojom se ptica kreće i ovdje je kao output prosljeđujemo ptici. Ukoliko je ptica pak prošla baš kroz sredinu objekta pipa, tada znači da joj je brzina kretanja trebala biti 0 jer je baš ondje gdje treba biti. Čim je „velocityPtice“ bliži vrijednosti nule tim je ptica točnija, tj. njezin output je bliže očekivanom outputu.

Aktivacijska funkcija koju sam koristio naziva se tanH funkcija. Ona je vrlo slična Sigmoid funkciji po svojem obliku, ali razlika je što ona svodi rezultat izračuna na vrijednosti između -1 i 1, dok Sigmoid svodi na vrijednosti između 0 i 1. Razlog zašto sam koristio tanH funkciju za razliku od Sigmoid funkcije je taj što sam trebao da mi rezultati idu i u pozitivne i negativne vrijednosti, kako bih mogao pomicat pticu i u pozitivnom, ali i u negativnom smjeru po Y osi. TanH se računa kao:

$$\tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

9. Zaključak

Kroz ovaj rad naučio sam mnogo o umjetnoj inteligenciji, a pogotovo o umjetnim neuronskim mrežama. Smatram kako je to jedna od vrlo važnih grana IT sektora koja će u budućnosti obuhvatiti većinu stvari koje trenutno radi čovjek. Mislim da sam uspješno iskoristio top – down pristup u prikazivanju sveobuhvatnog pojma umjetne inteligencije prodiranjem u svaku od njenih grana. Spomenuo sam i machine learning koji je dio umjetne inteligencije, ali se ne bi smio poistovjećivati s njom, jer je AI puno širi spektar. Nadalje sam prošao kroz grane ANN-a, tj. umjetnih neuronskih mreža koje su trenutni vrhunac umjetne inteligencije. Drago mi je da sam odabrao NEAT algoritam kao temeljni algoritam za ovaj rad jer je to još uvijek neistraženo područje. Na njemu se svakodnevno radi i smatram da je definitivno najzanimljivije područje baš zbog aspekta što je baziran na načelima prirodne evolucije. Ovaj pristup na neki način skreće pažnju s ljudskog mozga i stavlja ju na evoluciju živih bića što bi ponovno mogla biti prekretnica razmišljanja o umjetnoj inteligenciji. Smatram da je uključivanje matematičkih formula i grafova trebalo pridonijeti lakšem razumijevanju rada. Isto tako i korištenje usporedbi trebalo bi pobuditi razmišljanje o tome što sve još može ili što bi mogla napraviti umjetna neuroevolucija. Drago mi je da sam uspio napraviti jednu umjetnu neuronsku mrežu i prikazati djelić širokog spektra umjetne inteligencije ma koliko ona jednostavnija bila u usporedbi sa onim koje se danas koriste. Osim problema izrade umjetnih neuronskih mreža postoji i problem njihovog treniranja. Potrebno je biti pažljiv prilikom biranja pristupa problemu. Bitni su inputi, bitne su težine i njihov utjecaj na inpute. Samo prilagođavanjem neuronske mreže kada je jednom „kompletna“ može se doći do sve boljih i boljih rezultata. Smatram da je najbitnije od svega odabrati pripadajuću aktivacijsku funkciju ovisno o problemu s kojim se susrećemo i ovisno o tome što želimo da nam bude output mreže. Da li je to nešto linearno, varirajući, ili nešto jednostavno kao 1 ili 0, za sve postoji prikladna aktivacijska funkcija. Isto tako i prilikom odabiranja populacije, epoha treniranja, brzine skoka i ostalih aspekata treba paziti, prilagođavati i sve se u globalu svodi na pokušaj i pogrešku.

Popis literature

[1] Seth, H., SethBling (13.06.2015.) *MarI/O – Machine Learning for Video Games* [Video file]. preuzeto 04.10.2017. s https://www.youtube.com/watch?v=qv6UVOQ0F44&t=1s

[2] Unity Technologies (2018). *Unity Real-Time Developement Platform* (verzija 2018.3.9f1 Personal) preuzeto s https://store.unity.com/download?ref=personal

[3] Microsoft Corporation (2017). *Microsoft Visual Studio 2017* (verzija 15.9.28307.518) preuzeto s https://visualstudio.microsoft.com/downloads/

[4] Aron, G. (2019). *A* Pathfinding Project* (verzija 4.2.8.) preuzeto s https://arongranberg.com/astar/download

[5] Jethrel (16.04.2012.) *Castle Platformer Tiles* preuzeto s https://opengameart.org/content/castle-platformer

[6] Spencer, K. i Peter M. (11.08.2018.) *Gimp – GNU Image Manipulation Program* (verzija 2.10.8) preuzeto s https://www.gimp.org/downloads/

[7] Thorbjørn, L. (2008.-2018). *Tiled Map Editor* (verzija 1.2.4) preuzeto s https://thorbjorn.itch.io/tiled

[8] Sean, B. (2018). *Super Tiled2Unity* (verzija 1.5.1) preuzeto s https://seanba.itch.io/supertiled2unity/devlog/87454/supertiled2unity-version-151

[9] Khan Academy (2016). *Khan Academy courses* preuzeto s https://www.khanacademy.org/

[10] Herbert A. S. (1995). Artificial Inteligence: an empirical science. Department of Psychology, Carnegie Mellon University, Pittsburgh, 95-127.

[11] *Neuron SNARC stroja* [Slika] (bez dat.) preuzeto 10.07.2019. s http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/

[12] Weizenbaum , J. (sječanj 1966). Eliza – A Computer Program For the Study of Natural Language Communications Between Man and Machine. *Communications of the ACM*, Vol.9 Nr. 1, 36-35.

[13] Cmurobotics (1986). *NavLab1 (1986): Carnegie Mellon: Robotics Institute History of Self-Driving Cars* [Video file]. preuzeto 10.07.2019. s https://www.youtube.com/watch?v=ntIczNQKfjQ

[14] Baluja, S. (1996). Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics,* Vol. 26, No.3, 450-463.

[15] *Kretanje duhova u Pacman-u* [Slika] (2010) preuzeto 10.07.2019. s https://gameinternals.com/understanding-pac-man-ghost-behavior

[16] Birch C. (2010). Understanding Pac-Man Ghost Behavior. *GameInternals All theory, no practice*, preuzeto 10.07.2019. s https://gameinternals.com/understanding-pac-man-ghost-behavior

[17] Anderson, M. R. (bez dat.) *Twenty years on from Deep Blue vs Kasparov: how a chess match started the big data revolution* [Blog post]. Preuzeto 10.07.2019. s http://theconversation.com/twenty-years-on-from-deep-blue-vs-kasparov-how-a-chess-match-started-the-big-data-revolution-76882

[18] *Pojednostavljen prikaz modela Tree Search problema* [Slika] (2013) preuzeto 11.07.2019. s http://stanford.edu/~cpiech/cs221/apps/deepBlue.html

[19] Xu ,S. (bez dat.) *History of AI design in video games and its development in RTS games* [Blog post]. Department of Interactive Media & Game development, Worcester Polytechnic Institute. Preuzeto 14.07.2019. s https://sites.google.com/site/myangelcafe/articles/history_ai

[20] Su, H., Wang, X., Lin, Z., (2009). Flocking of Multi-Agents With a Virtual Leader. *Institute of Electrical and Electronics Engineers* Vol. 52(2), 293-307.

[21] Poulton, E. C. (1994). *Behavioral Decision Theory: A New Approach.* Trumpington street, Cambridge, USA: Cambridge University Press

[22] Kline, M., Mashable (31.08.2017.) *Elon Musk's 'Dota 2' Experiment is Disturbing Esports in a Big Way – No Playing Field* [Video file]. preuzeto 15.07.2019. s https://www.youtube.com/watch?v=jAu1ZsTCA64

[23] The Alphastar team(2019). *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II* [Blog post]. DeepMind. Preuzeto 16.07.2019. s https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

[24] Ramesh, R. (2017). *What is Artificial Intelligence? In 5 minutes.* [Video file], preuzeto 24.08.2019. s https://www.youtube.com/watch?v=2ePf9rue1Ao

[25] Bothun, D., Lieberman, M., Rao, A. S. (2017). Bot.Me: A revolutionary partnership, How AI is pushing man and machine closer together. PwC Consumer Intelligence Series, 1-18.

[26] Pandey, P. (2017). *Understanding the Mathematics behind Gradient Descent* [Blog post]. Medium Corporation, Towards Data Science. Preuzeto 25.08. 2019. s https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

[27] Ester, M., Kriegel, H., Sander, J. Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Institute for Computer Science, *University of Munich, Oetingenstr. 67, Germany,* 226-231.

[28] *Example of density based clustering* [Slika](2019.) Google Developers. Preuzeto 25.08.2019. s https://developers.google.com/machine-learning/clustering/clustering-algorithms

[29] Karpathy, A. (2016). *Deep Reinforcement Learning: Pong from Pixels* [Blog post]. Andrey Karpathy blog. Preuzeto 25.08.2019 s http://karpathy.github.io/2016/05/31/rl/

[30*] Illustration of a neuron* [Slika] (2018) Baillot, D. University of Callifornia – San Diego, Medical press. Preuzeto 26.08.2019. s https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html

[31] Bre, F., Gimenez, J. M., Fachinotti, V. D. (2017). *Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks.* Energy and Buildings 158

[32] Dertat A. (2017). *Applied Deep Learning – Part 1: Artificial Neural Networks* [Blog post] Towards Data Science. Preuzeto 26.08.2019. s https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

[33] Deeplizard (2017). Machine Learning & Deep Learning Fundamanetals, *Activation Functions in Neural Networks explained* [Blog post]. Preuzeto 27.08.2019. s https://deeplizard.com/learn/video/m0pIlLfpXWE

[34] Sharma S. (2017). *Activation Functions in Neural Networks, Sigmoid, tanh, Softmax, ReLU, Leaky ReLU EXPLAINED !!!* [Blog post]. Towards Data Science. Preuzeto 27.08.2019. s https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[35] Veen, F. V., [Slika] (2016). Preuzeto 27.08.2019. sa https://www.asimovinstitute.org/author/fjodorvanveen/

[36] Ng, A. (15.12.2017). *Andrew NG - The State of Artificial Intelligence* [Video file]. Preuzeto 27.08.2019. s https://www.youtube.com/watch?v=NKpuX_yzdYs

[37] Saha S., [Slika] (2018). *A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way.* Preuzeto 29.08.2019. s https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[38] Dasgupta T., [Slika](2019). *Deep Autoencoders using Tensorflow*. Preuzeto 29.08.2019. s https://towardsdatascience.com/deep-autoencoders-using-tensorflow-c68f075fd1a3

[39] *The two types of structual mutation in NEAT* [Slika] (2002). Stanley, K. O., Miikkulainen, R., Evolving Neural Networks through Augmented Topologies. Evolutionary Computation 10(2): 99-127

Popis slika