UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

GLOBOKO SPODBUJEVALNO UČENJE PROTI ČLOVEKU V IGRANJU RAČUNALNIŠKIH IGER

JOVANA MARKOVSKA

2022

ZAKLJUČNA NALOGA

MARKOVSKA

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Globoko spodbujevalno učenje proti človeku v igranju računalniških iger

(Deep reinforcement learning against human in playing video games)

Ime in priimek: Jovana Markovska
Študijski program: Računalništvo in informatika
Mentor: izr. prof. dr. Matjaž Kljun
Somentor: asist. dr. Domen Šoberl

Koper, september 2022

Ključna dokumentacijska informacija

Ime in PRIIMEK: Jovana MARKOVSKA

Naslov zaključne naloge:

Globoko spodbujevalno učenje proti človeku v igranju računalniških iger

Kraj: Koper

Leto: 2022

Število listov: 33 Število slik: 10

Število tabel: 3 Število referenc: 20

Mentor: izr. prof. dr. Matjaž Kljun

Somentor: asist. dr. Domen Šoberl

Ključne besede: spodbujevalno učenje, q-učenje, globoko učenje, globoke nevronske mreže, konvolucijske nevronske mreže

Izvleček:
Spodbujevalno učenje je področje strojnega učenja, ki se v zadnje čase zelo hitro razvija in se uporablja za zapletene, kompleksne naloge, kot so igre Atari. Namen diplomske naloge je raziskati, kako hitro lahko spodbujevalno učenje doseže raven človeškega igranja.

Izbrana je igra Pong za igralno platformo Atari 2600. V diplomski nalogi smo v programskem jeziku Python implementirali algoritem globokega spodbujevalnega učenja DQN, pri čemer smo skupaj s simulatorjem igralne platforme Atari 2600 uporabili knjižnico TensorFlow.

Za eksperimentalni del je bilo organizirano tekmovanje, na katerem so igralci igrali proti računalniku. Rezultate, ki so jih pri igranju igre dosegli človeški igralci, primerjamo z rezultati, ki jih je dosegla umetna nevronska mreža, naučena igrati proti računalniku po principu spodbujevalnega učenja. Ugotovili smo, da globoko spodbujevalno učenje zelo hitro doseže in preseže človeški nivo igranja igre.

Key document information

Name and SURNAME: Jovana MARKOVSKA

Title of the final project paper:

Deep reinforcement learning against human in playing video games

Place: Koper

Year: 2022

Number of pages: 33 Number of figures: 10

Number of tables: 3 Number of references: 20

Mentor: Assoc. Prof. Matjaž Kljun, PhD

Co-Mentor: Assist. Domen Šoberl, PhD

Keywords: reinforcement learning, q-learning, deep learning, deep neural networks,

convolutional neural networks

Abstract:
Reinforcement learning is a field of machine learning that has been developing very rapidly recently and is used for complex tasks. A popular use of reinforcement learning is to learn how to play video games. The purpose of the thesis is to investigate how quickly reinforcement learning can reach the level of human gaming.

We selected the Pong game on Atari 2600. We implemented the deep reinforcement learning algorithm DQN in the Python programming language using the TensorFlow library together with the emulator of the Atari 2600 gaming platform.

For the experimental part we organized a competition, where human players played a game against the computer. We compared the results achieved by human players with the results achieved by the artificial neural network, which learned to play the game against the computer through reinforcement learning. We find that deep reinforcement learning very quickly reaches and surpasses the human level of playing the game.

Zahvala

Rada bi se zahvalila svojima mentorjema za strokovno svetovanje, potrpežljivost in spodbudo pri nastajanju diplomskega dela.

Hvala tudi študentom, ki so mi pomagali izvesti tekmovanje.

Iskrena hvala tudi staršem za vso podporo in finančno pomoč pri študiju.

Hvala tudi vsem ostalim, ki ste mi vsa ta leta stali ob strani.

Kazalo vsebine

Kazalo tabel

Kazalo slik in grafikonov

1 Uvod

V zadnjih letih sta področji umetne inteligence in strojnega učenja z razvojem tehnologije in povečevanjem visokih računalniških zmogljivosti dobili zagon. Umetna inteligenca raziskuje kako lahko računalnik deluje bolj inteligentno, strojno učenje pa, kako iz danega nabora podatkov izluščiti znanje. Ker brez učenja ne moremo imeti inteligence, je področje strojnega učenja eno izmed ključnih področij umetne inteligence. Spodbujevalno učenje [17] je postalo zelo popularno področje stojnega učenja, ko se je leta 2013 algoritem globokega spodbujevalnega učenja *Deep Q-Network* naučil igrati videoigre na klasičnem sistemu Atari 2600 pod enakimi pogoji kot človek. To pomeni, da je kot vhodni podatek dobil zaslonsko sliko, kot izhodni podatek pa je vrnil akcijo na igralnem ploščku. Za uspešno igranje igre algoritem potrebuje vsaj milijon učnih korakov in veliko odigranih epizod.

Cilj te diplomske naloge je raziskati, kako dobro se algoritem DQN lahko kosa s človekom pri igranju videoiger za dva igralca, kjer je en igralec človek, drugi pa računalnik. Da bi odgovorili na to vprašanje, smo organizirali turnir igranja klasične igre Pong, ki se ga je udeležilo 18 študentov fakultete UP FAMNIT. Vsak od njih je odigral tri igre, rezultate pa smo primerjali z uspehom algoritma DQN pri igranju te iste igre proti računalniku ob različnih dolžinah učne faze. Delo zajema reimplementacijo spodbujevalnega učenja DQN v programskem jeziku Python, z uporabo knjižnice TensorFlow, v kombinaciji s simulatorjem arkadnih videoiger Atari 2600. Implementacija DQN vključuje programiranje globoke nevronske mreže, ki vsebuje več konvolucijskih plasti, katerih bistvo je obdelava vhodnih zaslonskih slik. Nevronska mreža tako lahko oceni koristnost neke akcije glede na trenutno stanje na zaslonu, koristnost akcij pa je osnovana na sistemu nagrajevanja, ki je v našem primeru vezan na točkovanje - dani gol pomeni pozitivno, prejeti gol pa negativno nagrado. To spodbuja agenta, da izbere zaporedje odločitev, ki mu bodo prinesli največjo skupno nagrado. Sistem na ta način po več sto odigranih igrah postopoma konvergira k neki uspešni strategiji igranja igre.

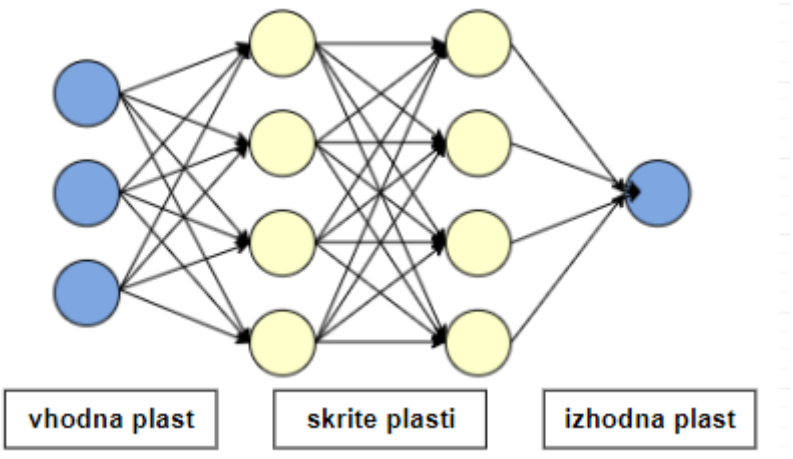
V diplomski nalogi najprej povzamemo teorijo globokega spodbujevalnega učenja, ki zajema dve pomembni področji strojnega učenja - globoke nevronske mreže, skupaj s konvolucijskimi mrežami ter Q-učenje. Sledi opis naše implementacije globokega spodbujevalnega učenja v programskem jeziku Python z uporabo knjižnice TensorFlow in simulatorja iger za sistem Atari 2600. Nato nadaljujemo s predstavitvijo izvedbe tekmovanja v igranju igre Pong in zaključimo z ugotovitvami naše raziskave in diskusijo odprtih vprašanj, ki lahko služijo kot podlaga za nadaljnje delo.

2 Globoko spodbujevalno učenje

Globoko spodbujevalno učenje združuje dve področji strojnega učenja - umetne nevronske mreže skupaj s konvolucijskimi mrežami ter spodbujevalno učenje. V tem poglavju povzamemo teoretične osnove obeh raziskovanjih področij ter teorijo globokega Q-učenja.

2.1 Umetne nevronske mreže

Nevronske mreže so modeli strojnega učenja, ki do neke mere posnemajo delovanje možganov. Možgani so sestavljeni iz nevronov, celic, ki prenašajo živčne impulze, podobno pa umetni nevroni v nevronskih mrežah prenašajo in pretvarjajo informacije med vhodnimi in izhodnimi signali. Nevronske mreže so ena od najpopularnejših metod strojnega učenja, ki se kot raziskovalna disciplina znotraj področja umetne inteligence ukvarja z vprašanjem, kako se računalniki lahko učijo samodejno iz razpoložljivih podatkov. Nevronske mreže torej ponujajo možnost reševanja problemov na načine, ki so bili razviti na podlagi razumevanja delovanja možganov in nevronskih povezav. Nevronske mreže torej na močno poenostavljen način simulirajo mehanizme učenja v bioloških sistemih. Definirane so kot skupina med seboj povezanih računskih enot – nevronov, in sicer tako, da je izhod enega nevrona povezan na vhod enega ali več drugih nevronov. Nevroni so organizirani v plasti, povezave med nevroni pa potekajo od vhodne plasti, preko skritih plasti, do izhodne plasti, kot je prikazano na sliki 1. Povezave med nevroni so utežene, vrednosti uteži pa določajo funkcijo, ki jo nevronska mreža izvaja. [2, 5]



Slika 1: Shematski prikaz enostavne umetne nevronske mreže.

2.1.1 Arhitektura umetnih nevronskih mrež

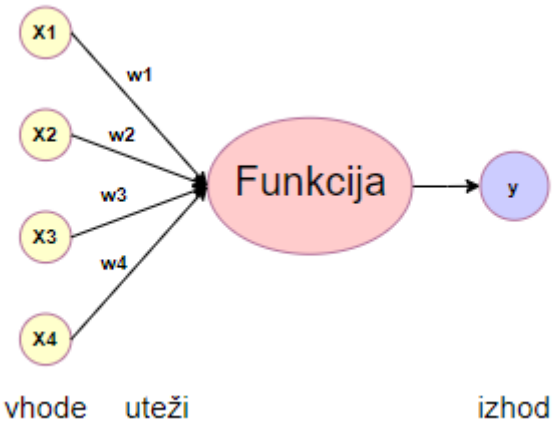
Obstaja veliko različnih vrst nevronskih mrež, ki se razlikujejo po kompleksnosti, namenu, arhitekturi, smeri pretoka podatkov in podobno. Za globoko učenje so tipične večplastne, globoke arhitekture, ki so običajno zmožne reševati kompleksnejše probleme kot nevronske mreže s plitko arhitekturo, pogosto pa vsebujejo tudi konvolucijske plasti, katerih namen je procesiranje slik. Glede na smer pretoka podatkov so najpogostejše napajalne (angl. feedforward) nevronske mreže, kjer se podatki premikajo med vozlišči samo v eno smer, vedno od vhodnih proti izhodnim komponentam in se nikoli ne vračajo k predhodnim plastem. Za razliko od njih, ponavljajoče (angl. recurrent) nevronske mreže delujejo tako, da podatke ohranjajo in jih vračajo na vhode, da bi tako izboljšale izhodni rezultat [6]. V diplomski nalogi obravnavamo in uporabljamo le napajalne nevronske mreže.

Nevronska mreža deluje tako, da preko vhodne plasti prejme vhodne podatke (najopogosteje vrednosti atributov obravnavanega primera), ti se nato širijo preko nevronskih povezav skritih plasti proti izhodnim nevronom ter odvisno od vrednosti uteži povezav in aktivikacijskih funkcij v nevronih, nekatere od teh izhodnih nevronov aktivirajo. Tako se vsak nabor vhodnih vrednosti preslika v nek določen nabor izhodnih podatkov. Aktivacija posameznega nevrona je odvisna od aktivnosti nevronov na predhodni plasti ter aktivacijske funkcije tega nevrona. Namen aktivacijske funkcije je vnesti nelinearnost na izhodu nevrona, kar nevronski mreži omogoča učenje nelinearnih funkcij [13, 15].

Izhodna vrednost nevrona y je izračunana kot:

$$y = f(\sum_{n=1}^{n} x_i \cdot w_i + b), \tag{2.1}$$

pri čemer so x_i vhodne vrednosti nevrona, w_i uteži na vhodnih povezavah, b prag proženja aktivacijske funkcije ter f aktivacijska funkcija (slika 2).



Slika 2: Delovanje umetnega nevrona.

Najpogosteje uporabljene aktivacijske funkcije so:

- *Sigmoidna funkcija*, ki uteženo vsoto vhodnih signalov preslika v interval (0, 1). Uporabna je predvsem v primerih, kjer napovedujemo verjetnosti.
- Funkcija *TanH* je podobna Sigmoidni funkciji, le da slika vhodne vrednosti v interval (-1, 1). Koristna je, kadar klasificiramo primere v dva razreda.
- Funkcija *ReLU (Rectified Linear Unit)* je aktivacijska funkcija, ki aktivira le določeno število nevronov naenkrat oz. ima deaktivirane nevrone, tam kjer je rezultat linearne transformacije enak 0. Zaradi svoje enostavnosti je zelo hitro izračunljiva in zato zelo popularna v globokih in konvolucijskih nevronskih mrežah.

2.1.2 Učenje nevronskih mrež

Nevronske mreže so univerzalni aproksimatorji funkcij, ki lahko z neko točnostjo aproksimirajo poljubno matematično funkcijo. To pomeni, da se lahko učijo katerekoli funkcije, neglede na njeno kompleksnost, točnost aproksimacije pa je odvisna od arhitekture mreže, narave učne množice podatkov ter števila iteracij učnih korakov. Kompleksnejše funkcije običajno zahtevajo več skritih plasti ter večje število nevronov na posamezni plasti.

Proces učenja nevronske mreže poteka preko prilagajanja uteži na učne podatke. Na začetku so uteži v nevronski mreži običajno nastavljene naključno. Nevronska mreža, ki še ni šla čez proces učenja, tako predstavlja neko naključno funkcijo. Napaka, ki jo bo takšna funkcija naredila pri preslikavi vhodnega podatka v izhodnega, bo, glede na želeni rezultat, relativno velika. Funkcijo, ki to napako določi oz. izmeri, imenujemo *funkcija izgube* (angl. *loss function*). Funkcija izgube je lahko definirana poljubno, pri učenju nevronskih mrež pa se pogosto uporablja *povprečna kvadratna napaka* (angl. *mean square error* ali *MSE*) ki jo uporabljamo tudi v tej diplomski nalogi:

$$\sum_{i=1}^{D}(x_i - y_i)^2 \tag{2.2}$$

Cilj učenja nevronske mreže je to napako minimizirati.

Najpogostejša metoda učenja nevronskih mrež je vzvratno razširjanje napake (angl. backpropagation). Ta metoda deluje tako, da izhodno napako po vsaki iteraciji učenja razširi od izhodov nazaj proti vhodom, pri tem pa uteži na vsaki vmesni plasti nekoliko prilagodi, in sicer tako, da se za trenutni učni primer napaka na izhodu nekoliko zmanjša. Nevronska mreža se po mnogih ponovitvah vzvratnega razširjanja napake na isti množici učnih primerov na te podatke prilagodi. Tipično je za to potrebno na stotine ali tisoče ponovitev. Pričakujemo pa lahko, da bo mreža dajala točne napovedi tudi na podatkih iz učne domene, ki jih med procesom učenja ni videla. Proces učenja nevronske mreže lahko povzamemo v sledečih korakih:

1. Vhodna plast sprejme enega ali več (sveženj) učnih primerov.
2. Učni primeri aktivirajo vhodne nevrone glede na trenutno nastavljene vhodne uteži.
3. Vhodni nevroni glede na trenutne vrednosti uteži skritih plasti aktivirajo skrite nevrone, ti pa nazadnje izhodne nevrone.
4. Funkcija izgube oceni napako izhodne plasti.
5. Algoritem vzvratnega razširjanja zmanjšuje napako z uravnavanjem uteži od izhodnih nazaj proti vhodnim nevronom.

Postopek ponavljamo, dokler ne dosežemo želene napovedne točnosti [19].

2.1.3 Konvolucija in konvolucijske nevronske mreže

Konvolucijske mreže so popularne pri problemih, ki vključujejo metode za obdelavo slik, kot so npr. razpoznavanje vzorcev (angl. *pattern recognition*) ali pridobivanje značilk (angl. *feature extraction*). Na področju obdelave slik je konvolucija proces uteženega seštevanja vrednosti sosednjih pik na sliki. Uteži so podane v majhni matriki, tipično velikosti 3 x 3 ali 5 x 5, ki se imenuje *jedro* (angl. *kernel*). Konvolucija vsako piko vhodne slike f(x, y) preslika v piko izhodne slike g(x, y) in sicer tako, da nanjo aplicira jedro $[\omega]$ na sledeči način [11]:

$$g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) \cdot f(x-dx, y-dy) \tag{2.3}$$

Z ustreznim jedrom lahko tako izvedemo različna filtriranja, kot npr. glajenje, ostrenje, detekcijo robov, itd. Takšna obdelava vhodne slike je bistvenega pomena za izluščenje značilk, ki jih lahko nadalje uporabimo pri strojnem učenju. Če bi, na primer, hoteli napisati program, ki bi na sliki prepoznaval oblike objektov, bi bil postopek detekcije robov na sliki preko konvolucije z ustreznim jedrom eden izmed prvih korakov v tem postopku. [8]

Globoko učenje avtomatizira iskanje jedra za dani problem obdelave vhodnih slik. Določanje ustreznih slikovnih značilk tako ni več v domeni programerja, ampak se jih nevronska mreža nauči sama. Takšne nevronske mreže so poznane kot konvolucijske nevronske mreže (angl. Convolutional Neural Network, CNN), za njih pa je značilno, da je vsaj ena izmed skritih plasti konvolucijska. [9, 14]

2.1.4 Arhitektura konvolucijskih mrež

Konvolucijska nevronska mreža je tipično sestavljena iz treh plast: konvolucijske, združevalne in polno povezane plasti [11]. Naloga konvolucjske plasti je učenje konvolucijskih filtrov

(oz. jeder). Arhitektura konvolucijske plasti pri tem določa naslednje parametre: število filtrov, velikost fitrov ter korak (angl. *stride*). Konvolucijska plast deluje tako, da vsak filter izvede konvolucijo po celotni vhodni sliki, pri tem pa se horizontalno in vertikalno premika z danim korakom. Velikost filtra in korak premikanja determinirata višino in širino izhodnih podatkov, število jeder pa njihovo globino. Izhod konvolucijskega filtra običajno vodi v združevalni sloj, ki zmanjša dimenzijo izhodnih podatkov tako, da združi sosednje vrednosti oz. izbere lokalni maksimum. To bistveno zmanjša število elementov, nad katerimi se učijo nadaljnje plasti, samo učenje pa je zaradi takšne abstrakcije tudi nekoliko bolj robustno. Nazadnje podatki potujejo še skozi polno povezano plast, ki služi kot povezava med konvolucijsko oz. združevalno plastjo in izhodno plastjo.

2.2 Spodbujevalno učenje

2.2.1 Kaj je spodbujevalno učenje

Bistvo spodbujevalnega učenja bomo najlažje razumeli na primeru vožnje kolesa [20]. Cilj je, da se voznik nauči voziti kolo, ne da bi padel. Po določenem času vožnje se voznik nagne v desno. Ko je nagnjen, ima dve možnosti: zaviti levo ali zaviti desno. Če voznik zavije levo, bo padel in tako spoznal, da zavijanje v levo ni pravilna odločitev, kadar je nagnjen v desno. Kasneje se voznik med vožnjo nagne v levo in spet ima dve možnosti: zaviti v levo ali v desno. Iz prejšnje nesreče se je naučil, da zavijanje levo ni dobra ideja, in meni, da je bolje zaviti desno. Ko zavije desno, se spet znajde na tleh in spozna, da tudi zavijanje v desno ni prava odločitev. Po določenem številu nesreč se voznik nauči, da zavijanje v levo ni dobro in obratno. Tako se na podlagi izkušenj naučil kako voziti, ne da bi padel.

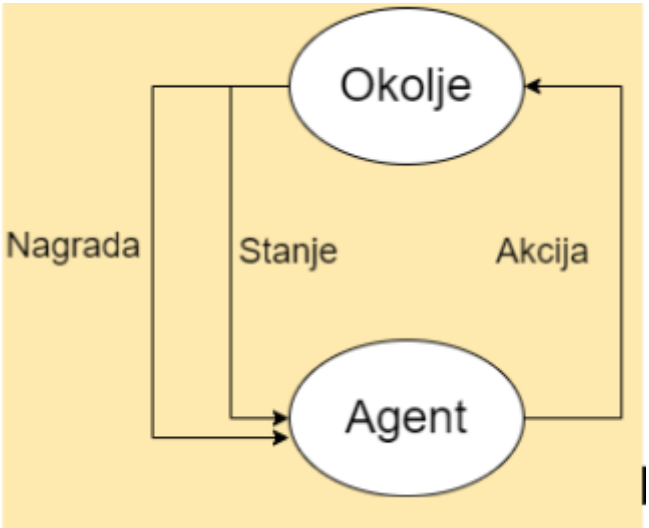
V spodbujevalnem učenju so dobra in slaba dejanja nagrajena oz. kaznovana. Cilj spodbujevalnega učenja je naučiti se strategije, ki maksimizira nagrado. V zgornjem primeru vsak padec s kolesa za voznika pomeni kazen oz. negativno nagrado.

Po Suttonovi in Bartovi definiciji [17] je spodbujevalno učenje področje strojnega učenja, ki preučuje probleme in načine njihovega reševanja. Učenje temelji na vprašanju, kako reagirati (oz. kakšne akcije izvesti) v določenih situacijah. Najzahtevnejši del spodbujevalnega učenja je dejstvo, da sistem ne ve, kakšno zaporedje akcij bi pripeljalo do želenega rezultata, ampak se mora tega naučiti sam. Učenje pa poteka po principu poskušanja, preko mehanizma nagrajevanja, pri katerem je nagrada običajno odložena. To pomeni, da ni nujno nagrajena zgolj ena sama akcija, ampak zaporedje več akcij, ki sistem pripeljejo v stanje, ko je ustrezna akcija lahko nagrajena.

2.2.2 Osnovni koncepti spodbujevalnega učenja

Spodbujevalno učenje modeliramo kot interakcijo med agentom in okoljem, v katerem agent z izvajanjem akcij prehaja iz stanja v stanje in prejema nagrade (slika 3). Pri tem je pomembno izpostaviti naslednje pojme:

- *Agent* je aktivni del spodbujevalnega modela, ki z izvajanjem akcij prehaja med stanji. To je lahko robot, animirana figura v računalniški igri ali kakšen drug računalniški sistem, ki sprejema odločitve.
- *Okolje* je domena, znotraj katere agent deluje. Za robota je to fizično okolje, v katerem se nahaja, za akterja v računalniški igri animiran svet, v katerem se igra odvija, v splošnem pa je to nabor vseh možnih stanj in akcij sistema.
- *Stanje* je nabor vseh lastnosti agenta v nekem trenutku, kot npr. pozicija ali orientacija v prostoru, položaj lastnih komponent, hitrost, temperatura, itd.
- *Akcija* je dejanje, ki ga agent lahko izvede in s tem vpliva na spremembo stanja.
- *Nagrada* je ocena koristnosti akcije oz. zaporedja akcij, ki jo agent prejme iz okolja po izvedeni akciji.

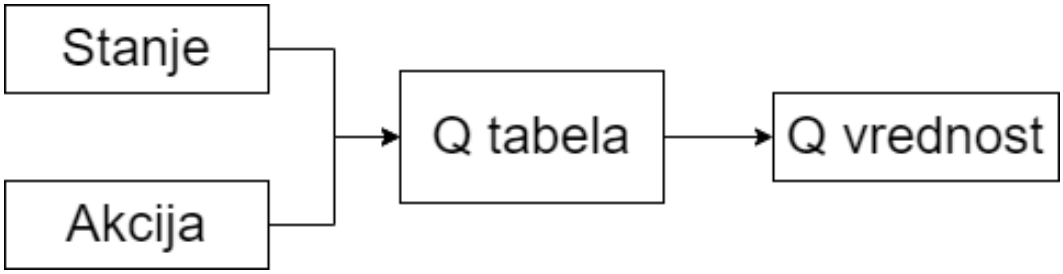


Slika 3: Model spodbujevalnega učenja.

Na sliki 3 je prikazana interakcija med agentom in okoljem. Agent opazuje okolje in na podlagi opažanj izvede akcijo, za katero v danem stanju predvideva, da bo prinesla najvišjo nagrado. Po izvedbi akcije od okolja prejme nagrado (ta je lahko pozitivna, negativna ali ničla) ter nova opažanja, ki mu povedo, kako je izvršena akcija vplivala na spremembo stanja. Na podlagi teh novih informacij lahko agent nato izboljša svojo strategijo in na ta način z izkušnjami napreduje, kar pomeni, da izboljšuje svoja predvidevanja o prihodnjih nagradah.

2.2.3 Q-učenje

Q-učenje je ena izmed najpopularnejših oblik spodbujevalnega učenja. Zanj je značilno, da se ne uči modela okolja, ampak zgolj koristnosti akcij. Od tod tudi ime Q-učenje, saj črka Q izhaja iz angleške besede *quality* (kvaliteta) in se nanaša na koristnost akcije v nekem stanju - višja Q vrednost pomeni koristnejšo akcijo. V svoji osnovni različici Q-učenje shranjuje Q vrednosti v dvodimenzionalni tabeli stanj in akcij. Ko se agent znajde v nekem stanju, v Q-tabeli enostavno preveri, katera od akcij v tem stanju ima najvišjo Q-vrednost ter to akcijo izvede. Ko po izvedbi akcije od okolja prejme nagrado, vrednosti v Q-tabeli prilagodi glede na ta nova opažanja in tako Q-tabela konvergira proti neki uspešni strategiji izvajanja akcij [16, 12].



Slika 4: Q-učenje s pomočjo Q tabele.

Proces Q-učenja je torej iterativno izboljševanje Q-tabele. Takšen pristop s Q-tabelo (slika 4) pa je možen samo v diskretnem okolju, kjer je stanj končno mnogo. V zveznem okolju Q-tabelo ponavadi zamenjamo s Q-funkcijo oblike Q(stanje, akcija) = Q vrednost. Te funkcije se mora agent seveda naučiti, zato jo modeliramo z nevronsko mrežo.

Algoritem Q-učenja deluje takole:

1. Inicializacija začetnih Q-vrednosti (npr. naključno ali vse na 0).
2. Ponovitev za n učnih korakov:
   - 2.1. Agent glede na trenutno stanje izbere akcijo z najvišjo Q-vrednostjo.
   - 2.2. Agent izvede akcijo in opazuje njen učinek (novo stanje, nagrada).
   - 2.3. Izračun nove Q-vrednosti prejšnjega stanja glede na izvedeno akcijo in prejeto nagrado.
   - 2.4. Posodobitev Q-funkcije.

Učno zanko ponavljamo tolko dolgo, dokler Q-vrednosti v povprečju rastejo oz. dokler ne dosežemo pričakovane učinkovitosti agenta. Zadnji korak zgornjega algoritma je posodobitev Q-funkcije glede na izvedeno akcijo in prejeto nagrado v nekem stanju. To posodobitev izvedemo z Bellmanovo enačbo:

$$\text{newQ}(s,a) = Q(s,a) + \alpha \cdot (R(s,a) + \gamma \cdot \max Q'(s',a') - Q(s,a)) \tag{2.4}$$

Nova Q-vrednost se torej izračuna tako, da se trenutni Q-vrednosti za izvedeno akcijo a v stanju s doda razlika med obstoječo in novo izračunano vrednostjo, pomnoženo s faktorjem α, ki predstavlja stopnjo učenja. Na ta način je učenje postopno in zato bolj stabilno kot če bi prejšnjo vrednost v celoti nadomestili z novo. Ta nova vrednost pa je sestavljena iz vsote takošnje nagrade r(s, a) ter najvišje Q-vrednosti maxQ'(S', a'), ki jo je mogoče dobiti v naslednjem stanju s' za katerokoli akcijo a'. Vrednost maxQ'(S', a') je pomnožena s faktorjem γ, ki vrednost odložene nagrade nekoliko zniža (angl. *discount factor*). Posledično algoritmu takojšnja nagrada pomeni nekoliko več kot odložena nagrada.

2.2.4 Globoko Q-učenje

Globoko Q-učenje je spodbujevalno Q-učenje, ki Q-funkcijo aproksimira z globoko oziroma konvolucijsko nevronsko mrežo. To mu omogoča, da se uči na zelo kompleksnih vhodnih podatkih, v našem primeru na zaslonskih slikah računalniške igre. Za razliko od tradicionalnega Q-učenja z uporabo Q-tabele ali enostavnejše nevronske mreže, globoko Q-učenje ne zahteva predhodne obdelave vhodne slike oziroma izločitve značilk (angl. *feature extraction*), ampak se nevronska mreža sama nauči ustreznih konvolucijskih filtrov. Tako lahko globoko Q-učenje računalniško igro igra pod enakimi pogoji kot človek — na podlagi tega, kar je vidno na zaslonu.

Učenje po principu spodbujevalnega učenja v vsakem učnem koraku proizvede eno izkušnjo. Izkušnja je v našem primeru prehod med dvema zaporednima zaslonskima slikama ob izvedeni akciji ter nagrada, ki jo agent prejme za to akcijo. Če bi nevronsko mrežo trenirali z zaporedjem izkušenj, kot jih agent dobiva med igranjem igre, Q-funkcija ne bi konvergirala k neki stabilni strategiji igranja, ampak bi se prilagajala vedno novejšim izkušnjam. Zato algoritem *globokega spodbujevalnega učenja* (*Deep Q-Networks*, DQN) [10] vse pridobljene izkušnje shrani v *pomnilnik izkušenj* (*replay memory*), ki služi kot vir učnih primerov. Po vsakem koraku spodbujevalnega učenja algoritem v pomnilnik izkušenj shrani novo pridobljeno izkušnjo, nato pa iz njega zajame naključno izbran sveženj (angl. *batch*) učnih primerov ter izvede eno iteracijo učenja nevronske mreže. Z vsakim korakom se torej množica učnih primerov veča, velikost svežnja pa je konstantna in tipično obsega 32 primerov. Ko pomnilnik izkušenj doseže neko končno velikost, npr. 1 milijon izkušenj, nove izkušnje pričnejo nadomeščati najstarejše shranjene izkušnje.

Algoritem globokega Q-učenja [10] lahko povzamemo na naslednji način:

1. Inicializiraj pomnilnik izkušenj s kapaciteto N učnih primerov.
2. Inicializiraj funkcijo Q (globoko nevronsko mrežo) z naključnimi utežmi.

3. Ponovi za M epizod:

    3.1. Inicializiraj začetno stanje igre.

    3.2. Za vsak korak znotraj trenutne epizode:

        3.2.1. Z verjetnostjo ϵ izberi akcijo a naključno, sicer glede na strategijo funkcije Q.

        3.2.2. Izvedi akcijo in pridobi nagrado ter opazuj naslednje stanje (zaslonsko sliko).

        3.2.3. Pridobljeno izkušnjo shrani v pomnilnik izkušenj.

        3.2.4. Iz pomnilnika izkušenj zajemi naključen sveženj učnih primerov.

        3.2.5. Posodobi funkcijo Q z uporabo Bellmanove enačbe (2.4).

3 Implementacija

Spodbujevalno učenje za igranje iger Atari smo implementirali v programskem jeziku Python z uporabo knjižnic *TensorFlow* [1] za implementacijo globokih nevronskih mrež, *OpenAI Gym* [4] v kombinaciji z arkadnim učnim okoljem ALE (Arcade Learning Environment) [3] za simulacijo igralne platforme Atari 2600 ter *Petting Zoo* [18], ki ponuja večagentni (angl. *multiagent*) vmesnik do okolja ALE, torej možnost igranja Atari iger za dva igralca. Vse naštete knjižnice so dostopne v uradnem Pythonovem repozitoriju knjižnic PIP.

3.1 Arkadno učno okolje

Arkadno učno okolje ALE emulira igralno konzolo Atari 2600 ter izvaja originalno binarno kodo (ROM) izbrane igre. Emulator v vsakem simuliranem časovnem koraku prejme diskretno akcijo igralnega ploščka, na izhod pa pošlje trenutno zaslonsko sliko v velikosti 160×210 pik. Akcije so oštevilčene z indeksi 0 – 17, mi pa uporabljamo zožen nabor akcij, ki so uporabljene v naši izbrani igri Pong. Te so: 0 (ni akcije), 1 (gumb strel), 2 (gumb gor) in 3 (gumb dol).

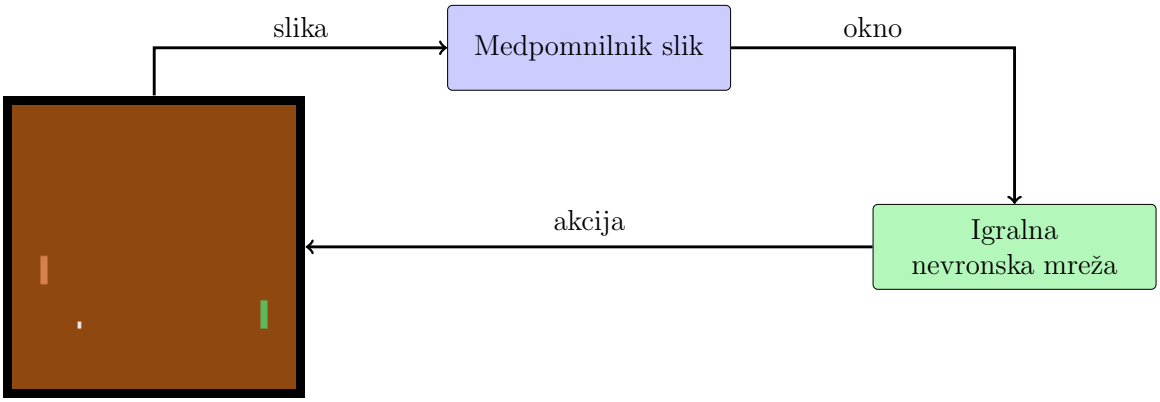
Knjižnica OpenAI Gym dodaja okolju ALE dodatno plast, ki poleg zaslonske slike vrne še informacijo o dobljeni nagradi (*reward*) ter o tem, ali se s trenutnim korakom kontinuiteta igranja prekine (*terminal*). Slednje se tipično zgodi takrat, ko igralec izgubi življenje in se stanje igre ponastavi ali pa ko je igra zaključena. Knjižnica Petting Zoo za igre z več igralci doda še informacijo o igralcu, ki je trenutno na vrsti.

3.2 Arhitektura nevronske mreže

Naloga naše nevronske mreže je, da na podlagi trenutnega stanja igre predlaga akcijo. Vhod v nevronsko mrežo je zaporedje štirih zaslonskih slik, s čimer zajamemo tudi del trenutne dinamike (smeri gibanja, hitrosti). Takšno zaporedje slik imenujemo *okno*. Zaslonske slike, ki jih prejmemo iz okolja ALE, pretvorimo iz barvnega formata RGB v 8-bitno sivinsko paleto, jih nekoliko obrežemo, tako da zajemajo le bistveni del dogajanja na zaslonu ter skrčimo na polovico velikosti, ker močno zmanjša vhodni prostor stanj, pri tem pa se zaradi zadostne ločljivosti slike ne izgubi nobena bistvena informacija o stanju igre. Vsaka tako obdelana slika je velikosti 84×84 pik.

Uporabili smo sledečo arhitekturo nevronske mreže, ki se je na podlagi eksperimentiranja z različnimi globinami in širinami izkazala za uspešno:

1. Vhodni sloj $4 \times (84 \times 84)$.



Slika 5: Interakcija med igralno nevronsko mrežo in arkadnim učnim okoljem ALE med igranjem igre.

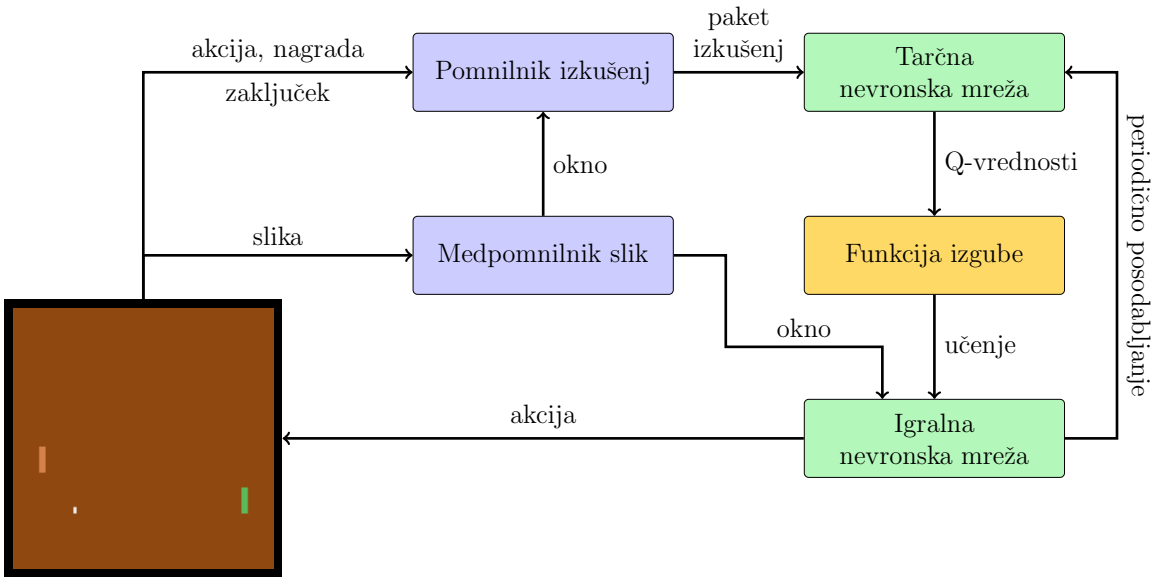
2. Konvolucijski sloj: 32 filtrov velikosti 8×8 s korakom $(4, 4)$.
3. Konvolucijski sloj: 64 filtrov velikosti 4×4 s korakom $(2, 2)$.
4. Konvolucijski sloj: 64 filtrov velikosti 3×3 s korakom $(1, 1)$.
5. Skriti sloj: 512 nevronov.
6. Izhodni sloj: 4 izhodi, ki pripadajo posameznim akcijam.

Na vseh nivojih, razen na izhodnem, uporabljamo aktivacijsko funckcijo ReLu, na izhodnem nivoju pa linearno aktivacijsko funkcijo. Dimenzije konvolucijskih slojev so uglašene z vhodnim formatom slik 84×84 pik. Prvi konvolucijski sloj z velikostjo filtrov 8×8 in korakom 4×4 takšno sliko skrči na velikost 20×20 pik. Druga konvolucijska plast z velikostjo filtrov 4×4 in korakom 2×2 proizvede slike velikosti 9×9 pik, kar je uglašeno s filtri velikosti 3×3 tretje konvolucijske plasti. Skriti sloj s 512 nevroni služi kot povezovalna plast med konvolucijskimi in izhodnimi sloji.

Takšna nevronska mreža vsaki od možnih štirih akcij pripiše realno število, ki predstavlja njeno Q-vrednost oz. koristnost. Algoritem nato izvede akcijo z najvišjo Q-vrednostjo. Interakcija med naučeno *igralno nevronsko mrežo* (angl. *Policy Neural Network*) in okoljem ALE je prikazana na sliki 5. Vsaka zaslonska slika, ki jo izvozi ALE, gre najprej v medpomnilnik, kjer se obreže in skrči ter združi s predhodnimi tremi slikami v okno. Igralna nevronska mreža nato na podlagi trenutnega okna določi akcijo.

3.3 Globoko spodbujevalno učenje

Preden lahko igralna nevronska mreža uspešno odigra igro, se mora igranja naučiti. Tehnično gledano to pomeni, da se mora nevronska mreža naučiti vhodna okna preslikati v takšne nabore Q-vrednosti, da izvajanje akcij z najvišjo Q-vrednostjo dolgoročno doseže čim več točk. Povratna informacija, ki je algoritmu na voljo, je vrednost nagrade, ki jo prejme po vsaki izvedeni akciji. V našem primeru je nagrada enaka 1, če nasprotniku damo gol, -1, če gol



Slika 6: Arhitektura globokega spodbujevalnega učenja DQN za učenje igranje igre na platformi Atari.

prejmemo ter 0 sicer. Velika večina akcij torej ne prejme nagrade (oz. prejme nagrado z vrednostjo 0), kar pomeni, da je nagrada (pozitivna ali negativna) precej odložena, algoritem pa mora izvesti dolgo zaporedje akcij, preden dobi informativno povratno informacijo. Spodbujevalno učenje je v takšnih primerih še posebej kompleksno, saj mora algoritem preiskati zelo velik prostor stanj, preden pride do informacij, ki mu kasneje omogočajo umerjati preiskovanje. Algoritem globokega spodbujevanega učenja DQN je prilagojen prav takšnim problemom.

Globoko spodbujevalno učenje smo zasnovali na način, ki je prikazan na sliki 6. Igralni zanki s slike 5 smo dodali še vzporedno, učno zanko. Ta zanka poleg trenutnega stanja (okna) obravnava tudi izvršeno akcijo, pridobljeno nagrado, stanje po izvedeni akciji ter informacijo o kontinuiteti stanj (končno). Vse te informacije skupaj predstavljajo *izkušnjo* (glej tabelo 7). Algoritem po vsakem koraku igranja igre pridobi novo izkušnjo, ta pa se shrani v pomnilnik izkušenj, od koder se neodvisno od trenutnega dogajanja v igri v vsakem koraku igranja naključno generira paket (angl. *batch*) izkušenj za učenje nevronske mreže. Velikost paketa smo nastavili na 32 izkušenj.

Za učenje nevronske mreže uporabimo funkcije izgube (angl. *loss function*):

$$\text{loss} = \left(y - Q_P(\text{okno}_i, \text{akcija}_i)\right)^2, \tag{3.1}$$

kjer je y tarčna Q-vrednost, izračunana na podlagi izkušenj, $Q_P(\text{okno}_i, \text{akcija}_i)$ pa Q-vrednost, kot jo predvideva trenutno stanje igralne nevronske mreže. Tarčno Q-vrednost y za posame-

okno i	akcija i	nagrada i	okno $i+1$	končno $i+1$
$4 \times (84 \times 84)$	$0-3$	$0, 1, -1$	$4 \times (84 \times 84)$	true/false

Slika 7: Struktura posamezne izkušnje.

zno izkušnjo izračunamo kot:

$$y = \text{nagrada}_i + \gamma \cdot \max(Q_T(\text{okno}_{i+1})), \tag{3.2}$$

kjer prihodnje Q-vrednosti $Q_T(\text{okno}_{i+1})$ za vse razpoložljive akcije izračuna *tarčna nevronska mreža*. Slednja je kopija *igralne nevronske mreže* in se periodično posodablja vsakih 10.000 korakov. Igralna nevronska mreža je torej vedno nekaj korakov pred tarčno nevronsko mrežo. Tako je predvidena prihodnja Q-vrednost je znižana za faktor γ, ki smo ga nastavili na $0,99$, s tem pa dali majhno prednost takojšnji nagradi pred odloženo nagrado.

Funkcijo izgube na koncu uporabimo za učenje igralne nevronske mreže na danem paketu izkušenj. To opravi knjižnica TensorFlow z vzvratnim širjenjem napake, pri čemer je namen minimizirati napako, kot jo definira funkcija izgube. Uporabili smo optimizacijski algoritem *Adam* [7] s stopnjo učenja (angl. *learning rate*) $\alpha = 2.5 \cdot 10^{-4}$.

3.4 Pomnilnik izkušenj

Ker imajo nevronske mreže nizko stopnjo učenja, morajo običajno večkrat iterirati skozi isto množico učnih primerov. Za ta namen DQN uporablja pomnilnik izkušenj, od koder se v vsakem koraku zajame naključen paket učnih primerov, sam pomnilnik pa nove primere prejema iz učnega okolja oz. v našem primeru iz arkdane simulacije. Zaželeno je, da je kapaciteta pomnilnika izkušenj čim večja, saj to zmanjša možnost pretirane prilagoditve (angl. *overfitting*) na nek specifičen nabor primerov. Vendar, kot je razvidno iz tabele 7, posamezna izkušnja zavzame relativno veliko pomnilniškega prostora, in sicer približno 55 KB. Če bi želeli izvesti milijon učnih korakov ter shraniti vse pridobljene izkušnje, bi tako potrebovali več kot 52 GB prostega pomnilnika. Vendar pa lahko z nekoliko premišljenim pristopom shranjevanja izkušenj prostorsko zahtevnost pomnilnika izkušenj bistveno zmanjšamo.

Pomnilnik izkušenj definiramo kot polje (angl. *array*) elementov sledečega tipa:

- akcija (vrednost $\{0, 1, 2, 3\}$, 1 zlog)
- nagrada (vrednost $\{0, 1, -1\}$, 1 zlog)
- okno (velikost 84×84 pik, 7056 zlogov)
- končno (vrednost true/false, 1 zlog)

Posamezen vnos tako obsega 7059 zlogov, za hranjenje enega milijona takšnih vnosov pa potrebujemo približno 6,5 GB prostega pomnilnika.

Naključno izkušnjo lahko poustvarimo iz zaporedja vnosov tako, da zajamemo pet zaporednih vnosov, in sicer takšnih, da noben od petih nima zastavice *končno* ali pa zgolj zadnji. Na ta način ohranimo kontinuiteto posamezne izkušnje. Iz prvih štirih zajetih vnosov nato poustvarimo okno i, iz zadnjih štirih pa okno $i + 1$. Akcijo, nagrado in zastavico *končno* vzamemo iz petega vnosa.

3.5 Preiskovanje prostora stanj

Opisana implementacija spodbujevalnega učenja igranja videoiger je sicer prilagojena za sistem Atari 2600, vendar pa je z vidika posamezne igre na tem sistemu domensko popolnoma neodvisna, kar pomeni, da ne vsebuje nobenega predhodnega znanja o pravilih igre ali uspešnih strategijah igranja. Pričakujemo lahko torej, da bodo začetne faze učenja zelo podobne naključnemu pritiskanju gumbov igralnega ploščka. Takšno naključno vedenje je v začetnih fazah učenja celo zaželeno, saj je koristno, da sistem pridobi čim bolj raznolike izkušnje. S tehničnega vidika to pomeni, da je preiskovanje prostora stanj neinformirano oz. neusmerjeno. V kasnejših fazah učenja, ko nevronska mreža že začne konvergirati proti neki strategiji igranja igre, pa pričakujemo, da bodo izbrane akcije vedno bolj smiselne in uspešne, preiskovanje pa vedno bolj usmerjeno v prefinjevanje izoblikovane strategije.

Razmerje med usmerjenim in neusmerjenim spodbujevanim učenjem (v strokovni literaturi se za to uporabljata angleška izraza *exploration* in *exploitation*) uravnavamo s parametrom $\epsilon \in [0, 1]$, pri čemer $\epsilon = 1$ pomeni popolnoma naključno preiskovanje (exploration), $\epsilon = 0$ pa igranje po naučeni strategiji brez naključnih potez (exploitation). Algoritem torej ob vsaki izbiri akcije z verjetnostjo ϵ akcijo izbere naključno. Učenje igre pričnemo z vrednostjo $\epsilon = 1$, ki jo linearno s številom učnih korakov zmanjšujemo proti vrednosti $\epsilon = 0.1$. Mi smo stopnjo zmanjševanja nastavili tako, da $\epsilon = 0.1$ dosežemo pri milijonu učnih korakov, od tam naprej pa je ta vrednost konstantna. Tudi ob uporabi dokočno naučenega modela dopuščamo 10 % naključno izbranih akcij, kar igri doda nekaj dinamike, še posebno, če se igralna epizoda vedno začne na enak način.

4 Tekmovanje v igranju igre Pong

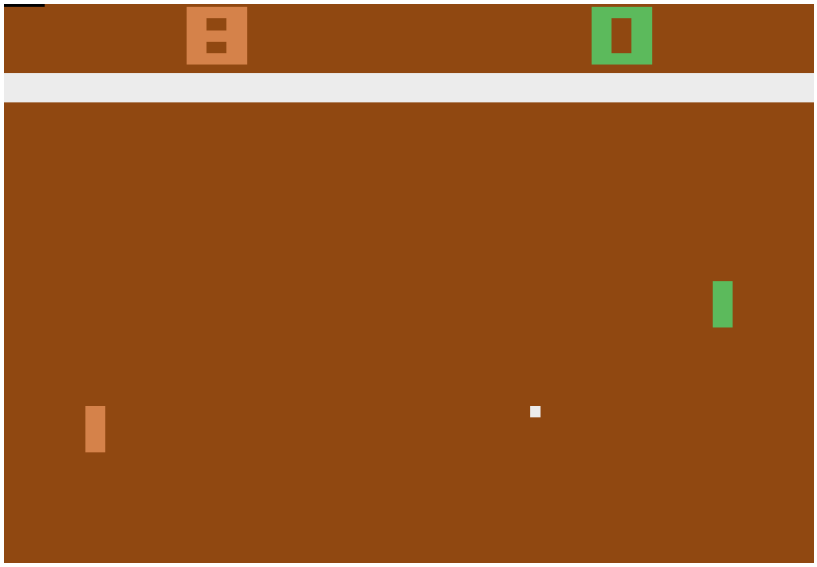
V tem poglavju je primerjana uspešnost človeka in globokega spodbujevalnega učenja v igranju igre Pong. Tekmovanji sta potekali ločeno - človek proti algoritmu ter nevronska mreža, naučena s spodbujevalnim učenjem, proti istemu algoritmu. Rezultate smo merili po enaki metriki in jih tako naredili primerljive.

4.1 Pravila igre

Pong je igra, pri kateri si človek in računalnik med seboj podajata majhno žogo. Igralec mora s prvim odbojem žogo odbiti na drugo stran ter ves čas predvidevati, kako bo nasprotnik odbil žogo. Če žoge ne odbijemo, dobimo gol, kar nasprotniku prinese točko. Zmaga igralec, ki prvi zbere 21 točk.

Pravila igre:

1. Udeleženca sta računalnik (vgrajeni algoritem v igri Pong) in igralec (človek oziroma nevronska mreža).
2. Igralec uporablja tipkovnico za premikanje loparja gor in dol ter za serviranje žogice. Nevronska mreža te ukaze pošilja simulatorju programsko.
3. Algoritem vedno igra na levi strani, človek oziroma nevronska mreža vedno na desni strani.

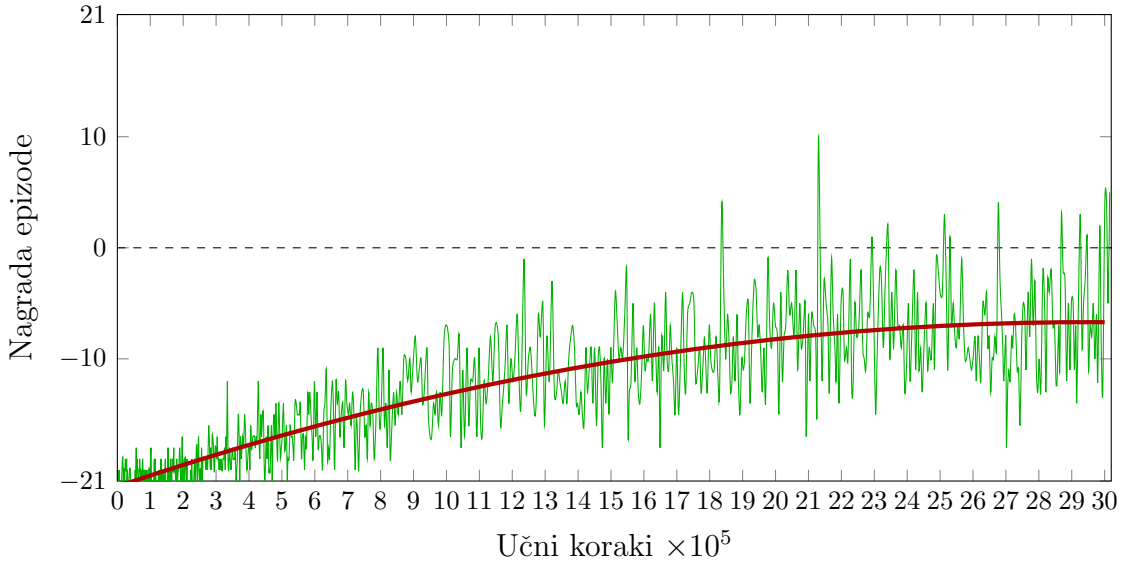


Slika 8: Zaslonska sloka igre pong.

Na sliki 8 na prikazan zaslon, kot ga vidi igralec med igro. Na levi strani igra računalnik, na desni igralec in v sredini je žogica. V zgornjem delu je trenutni rezultat.

4.2 Nevronska mreža proti računalniku

Spodbujevalno učenje igranja igre Pong smo izvajali 49 ur na osebnem računalniku s 6-jedrnim procesorjem Intel Xeon, ki teče s frekvenco 3.5 GHz. Možnosti poganjanja na GPU nismo imeli. V tem času je algoritem izvedel 3 milijone učnih korakov in tako odigral 797 iger (epizod) Ponga proti računalniku. Naučeni model smo shranili vsakih 100.000 korakov ter tako pridobili 30 različno usposobljenih nevronskih mrež.



Slika 9: Graf konvergence spodbujevalnega učenja igranja igre Pong v odvisnosti od števila učnih korakov. Zelena krivulja prikazuje višino nagrad, pridobljenih pri posamezni igri. Rdeča krivulja je izračunani trend.

Napredek pri učenju prikazuje graf na sliki 9. Zelena krivulja predstavlja višino nagrade, pridobljene po vsaki odigrani igri. Rdeča krivulja je izračunani trend nagrade. Nagrada epizode je definirana kot razlika med številom točk, ki jih je dosegel računalnik, in številom točk, ki jih je dosegla naučena strategija. Pozitivna nagrada torej pomeni, da je zmagala nevronska mreža, negativna, da je zmagal računalnik, ničla pa, da je bil rezultat neodločen. Nevronska mreža je računalnik prvič premagala v epizodi 568, po približno 29 urah učenja oz. 1,8 milijona učnih korakov z rezultatom 21 : 17. Po tem je zmagala še 13-krat, najvišji dosežen rezultat pa je bil 21 : 11.

4.3 Človek proti računalniku

Za primerjavo igranja med človekom in računalnikom smo organizirali turnir oz. tekmovanje. Tekmovanje je potekalo v učilnici na fakulteti UP FAMNIT na Sentorjevi ulici in je bilo izvedeno na prenosnem računalniku. Tekmovanja se je udeležilo 18 študentov. Vsak je odigral tri igre. Ena igra je trajala približno 15 minut, celotno tekmovanje pa tri dni, približno

štiri ure na dan. Igralci so igro opredelili kot težko in nemogočo za zmago[1]. Na začetku so bili igralci nekoliko zmedeni, saj je nasprotnik igral zelo dobro, medtem ko so se oni šele privajali na hiter potek igre. Sčasoma so se igralci navadili na igro, začeli so igrati stabilno in so prišli do faze, pri kateri so poskušali z različnimi triki doseči gol. Izkazalo se je, da je strategija igranja vgrajenega algoritma relativno enostavna in predvidljiva — računalnik enostavno sledi vertikalni poziciji žogice, gol pa lahko dosežemo tako, da žogico poženemo z višjo hitrostjo kot računalnik lahko premika svoj lopar.

[1]Eden od razlogov je verjetno dejstvo, da je bila igra Pong na Atari 2600 narejena za igranje s posebnim analognim kontrolerjem, pri katerem lopar premikamo z vrtenjem koleščka. Igranje preko tipkovsnice je nekoliko težje.

5 Rezultati

V nadaljevanju so prikazani rezultati vseh odigranih iger turnirja. V analizo je bilo zajetih 54 iger ter 18 igralcev.

Tabela 1: Rezultati igranja udeležencev turnirja proti računalniku.

Igralec	Igra 1	Igra 2	Igra 3	Igralec	Igra 1	Igra 2	Igra 3
1	21:3	21:5	21:11	10	21:1	21:4	21:5
2	21:4	21:6	21:6	11	21:8	21:17	21:17
3	21:5	21:8	21:10	12	21:7	21:2	21:12
4	21:2	21:5	21:11	13	21:4	21:6	21:9
5	21:1	21:1	21:2	14	21:2	21:2	21:6
6	21:1	21:6	21:10	15	21:7	21:10	21:8
7	21:6	21:5	21:4	16	21:4	21:9	21:10
8	21:2	21:1	21:10	17	21:8	21:11	21:7
9	21:5	21:9	21:11	18	21:2	21:6	21:15

V tabeli 1 so prikazane točke računalnik : človek za vseh 54 iger. Igralce smo označili s številkami od ena do 18. Vidimo lahko, da je pri vseh 54 igrah zmagal računalnik, igralci pa so se po uspehu zelo razlikovali.

Razlika med danimi in prejetimi goli je enaka nagradi epizode, kakršno spodbujevalno učenje prejme med treniranjem. Rezultati iger, pretvorjeni v epizodne nagrade, so prikazani v tabeli 2. Igralci so v povprečju dosegli epizodno nagrado -14,54, kar je spodbujevalno učenje doseglo po približno 800.000 učnih korakih, kot je razvidno z grafa na sliki 4.2.

Število doseženih golov pa nam ne pove dovolj o napredku pri igranju igre. Če gol prejememo zelo hitro, je to slabše, kot če gol prejememo po tem, ko smo gol že nekaj časa uspešno branili. In če gol damo hitro, je to bolje, kot če ga damo po tem, ko se je računalnik že nekaj časa uspešno branil. Zato za primerjavo rezultatov uvajamo metriko, ki vključuje tudi dolžino vsake odigrane epizode. Igralce smo ocenili po enačbi

$$\text{ocena} = \frac{\sum_{i=1}^{n} 10^3 \cdot r_i \cdot d_i^{-1}}{n}, \tag{5.1}$$

kjer je $r_i \in \{1, -1\}$ nagrada, ki jo je igralec prejel po končani $i - ti$ epizodi, d_i dolžina epizode in n število epizod v odigrani igri. Dolžina epizode je merjena v številu korakov oz. akcij, hitrost igre pa je potekala z 20 koraki na sekundo. Ocene vseh 54 iger so prikazane v tabeli 3.

Iz tabele je razvidno, da so igralci napredovali s časom, saj so v povprečju pri vsaki nadaljni igri dosegli boljši rezultat. Povprečna ocena iger igralcev, skupaj s standardnim odklonom, predstavlja interval uspešnosti, ki so ga dosegli študenti na našem tekmovanju. Da bi ocenili, kako hitro lahko spodbujevalno učenje doseže in preseže ta nivo, je nevronska

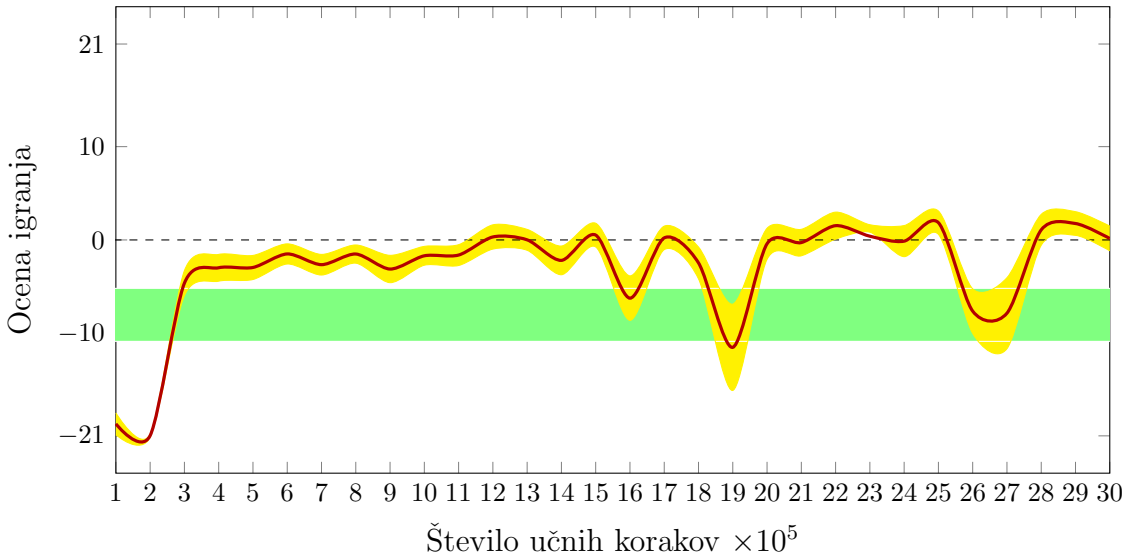
Tabela 2: Epizodna nagrada igralca po vsaki odigrani igri.

Igralec	Igra 1	Igra 2	Igra 3	Povprečje
1	-18	-16	-10	-14.66
2	-17	-15	-15	-15.66
3	-16	-13	-11	-13.33
4	-19	-16	-10	-15.00
5	-20	-20	-19	-19.66
6	-20	-15	-11	-15.33
7	-15	-16	-17	-16.00
8	-19	-20	-11	-16.66
9	-16	-12	-10	-12.66
10	-20	-17	-16	-17.66
11	-13	-4	-4	-7.00
12	-14	-19	-9	-14.00
13	-17	-15	-12	-14.66
14	-19	-19	-15	-17.66
15	-14	-11	-13	-12.66
16	-17	-15	-11	-13.33
17	-13	-10	-14	-12.33
18	-19	-15	-6	-13.33
Povprečje	-17.00	-14.72	-11.89	-14.54
Standardni odklon	2.43	4.00	3.74	2.74

mreža po vsakih 100.000 korakih učenja odigrala 100 iger proti računalniku, kar je skupaj 3000 iger. Povprečna ocena vsake faze igranja, skupaj s standardnim odklonom, je prikazana na grafu slike 10. Interval uspešnosti človeka je bil -8,08, s standardnim odklonom 2,84, kar je nevronska mreža presegla že po 300.000 učnih korakih.

Tabela 3: Ocene udeležencev turnirja izračunane po enačbi (5.1).

Igralec	Igra 1	Igra 2	Igra 3	Povprečje
1	-7.72	-9.92	-5.78	-7.81
2	-12.94	-8.74	-7.73	-9.80
3	-7.71	-6	-5.67	-6.46
4	-12.45	-10.01	-3.73	-8.43
5	-16.42	-15	-13.2	-14.87
6	-12.9	-7.36	-5.64	-8.63
7	-6.93	-7.49	-9.59	-8.00
8	-14.72	-10.23	-5.31	-10.09
9	-7.41	-5.67	-3.03	-5.37
10	-14.46	-9.39	-6.37	-10.07
11	-4.96	-2.39	-2.02	-3.12
12	-7.30	-8.49	3.92	-6.57
13	-9.66	-7.06	-5.92	-7.55
14	-14.58	-14.44	-10.06	-13.03
15	-5.54	-4.69	-6.60	-5.61
16	-9.90	-4.72	-4.41	-6.34
17	-5.28	-3.96	-6.49	-5.24
18	-11.55	-5.84	-4.28	-7.22
Povprečje	-10.14	-7.86	-6.10	-8.03
Standardni odklon	3.69	3.35	2.71	2.84



Slika 10: Primerjava igranja igre Pong med naučeno nevronsko mrežo in človekom. Rdeča krivulja prikazuje povprečno oceno 100 odigranih iger nevronske mreže po n učnih korakih; standarni odklon je označen z rumenim pasom. Zeleni pas prikazuje oceno igranja ljudi znotraj standardnega odklona od povprečja.

6 Diskusija

Ugotavljali smo, kako hitro lahko globoko spodbujevalno učenje doseže človeški nivo igranja preproste računalniške igre za dva igralca. Izbrali smo igro Pong na platformi Atari 2600. Implementirali smo algoritem globokega spodbujevalnega učenja DQN, in sicer v programskem jeziku Python z uporabo knjižnice *TensorFlow*. Uporabili smo učno okolje *ALE* skupaj s knjižnico *OpenAI Gym* za simulacijo igralne platforme Atari 2600 ter omogočili igro za dva igralca s pomočjo knjižnice *PettingZoo*. Spodbujevalno učenje igranja igre Pong je trajalo dva dni. V tem času je izvedlo 3 milijone učnih korakov oz. odigralo 797 iger. Organizirali smo tudi tekmovanje, ki se ga je udeležilo 18 študentov. Vsak igralec je odigral tri igre. Tekmovanje je trajalo 3 dni.

Rezultati tekmovanja so pokazali, da globoko spodbujevalno učenje človeškega igralca lahko preseže po približno 800.000 učnih korakih, če upoštevamo zgolj končni rezultat igre. Po oceni uspešnosti igranja, ki zajema tudi dolžino epizode, pa računalnik lahko doseže človeški nivo igranja že po približno 300.000 učnih korakih. Od tod lahko sklepamo, da se spodbujevalno učenje relativno hitro nauči ubraniti golov, potrebuje pa nekaj več časa, da se nauči dati gol nasprotniku.

Igralci, ki so se udeležili tekmovanja, so bili v igranju igre Pong začetniki. Med prvim igranjem igre so se šele privajali na kontroliranje svojega loparja ter dinamiko igre, strategijo igranja pa so začeli razvijati šele med drugim ali tretjim igranjem. Če bi želeli ugotavljati, kako ocena človeških igralcev konvergira s časom, bi morali izvesti bistveno daljši eksperiment, ki bi lahko trajal celo več tednov. Druga možnost bi bila povabiti igralce, ki igro Pong na tem sistemu že obvladajo, vendar gre za več desetletij staro igro, zato bi bilo takšne igralce težko ali celo nomogoče dobiti.

Igralcem se je igra zdela težka ne samo zato, ker so jo igrali prvič, ampak tudi zato, ker je bila izvorno narejena za uporabo z analognim kontrolerjem, pri katerem igralec lopar premika z vrtenjem koleščka. Nadomestitev takšnega principa kontrole z navadno tipkovnico igranje nekoliko oteži, saj je natančno nastavljanje položaja loparja preko diskretnih ukazov človeku manj naravno. Težko je predvideti, koliko bi se rezultati igranja pri ljudeh izboljšali z uporabo analognega kontrolerja, vendar zelo verjetno ne dovolj, da bi lahko presegli nivo, ki ga je doseglo spodbujevalno učenje.

Odprto ostaja vprašanje, zakaj je spodbujevalno učenje konvergiralo pri oceni okoli ničle, kar pomeni izenačitev z računalniškim igralcem oz. vgrajenim algoritmom igranja. Ni težko ugotoviti, da je premikanje loparja skladno z višino žogice optimalna defenzivna strategija. Če lopar lahko premikamo enako hitro kot se premika žogica, igralcu, ki uporablja takšno strategijo, nikoli ne moremo dati gola. Vgrajeni algoritem uporablja takšno strategijo, vendar je hitrost njegovega loparja omejena, zato lahko prejme gol, kadar žogico z visoko hi-

trostjo usmerimo daleč stran od njegovega trenutnega položaja loparja. Kljub temu je strategija vgrajenega algoritma precej blizu optimalni defenzivni strategiji, zato je igra tako za igralce kot za spodbujevalno učenje skoraj nepremagljiva. Sklepamo lahko, da bi spodbujevalno učenje pomanjkljivost počasnega loparja računalniškega igralca na neki točki odkrilo in začelo izkoriščati, vprašanje pa je, koliko učnih korakov bi bilo za to potrebnih. Ta raziskava je pokazala, da zagotovo več kot 3 milijone.

Zanimiva smer za nadaljnje delo je izvedba tekmovanja človeka neposredno proti nevronski mreži. Tehnično za takšno izvedbo igranja ni nobenih ovir, jasno pa je, da je dolžina igranja, ki jo spodbujevalno učenje za to igro zahteva, za človeka predolga. Ne moremo namreč pričakovati, da bo človek odigral več sto ali tisoč epizod proti računalniku, tudi če bi imel vmes možnost počitka. Bolj smiseln pristop je nevronsko mrežo natrenirati ločeno ter jo potem uporabiti proti človeku. To smo sicer preizkusili, vendar se je izkazalo kot neuspešno, saj se nevronska mreža preveč prilagodi računalnikovi defenzivni strategiji in se kasneje proti človeškemu igralcu vede skoraj kot popoln začetnik. Človek namreč svoj lopar premika bistveno drugače kot računalnik — običajno počaka, da žogica pride bliže ter ga nato v kratkem zamahu poskuša premakniti k žogici. Računalnik, po drugi strani, loparja nikoli ne drži na mestu in s počasnim premikanjem sledi žogici. Takšna bistvena sprememba v nasprotnikovi strategiji nevronsko mrežo močno “zmede”. V fazi učenja namreč nikoli ni pridobila izkušnje takšne hitrosti nasprotnika, niti izkušnje njegovega mirovanja. Znajde se v situaciji, ko se mora praktično začeti učiti znova.

Rešitev tega problema morda leži v drugačnem pristopu k treniranju nevronske mreže. Namesto igranja z računalnikom bi se med seboj pomerili dve nevronski mreži, vsaka na svoji strani igralnega polja. To bi bistveno povečalo prostor stanj, ki ga mora spodbujevalno učenje preiskati. Jasno je namreč, da v takšni konfiguraciji prostor zajema vse možne nasprotnikove strategije igranja igre, od najbolj nesmiselnih in neuspešnih, do optimalnih. Poraja se vprašanje, če bi spodbujevalno učenje našlo kakšno od uspešnih strategij v doglednem času in če se ne bi pri tem zopet ujeli v enako past — mreži bi se prilagodili neki iznajdeni strategiji, po kateri bi bil povprečni rezultat verjetno približno neodločen, ob zamenjavi nasprotnika s človeškim igralcem pa bi se nevronska mreža morala spet začeti prilagajati novi strategiji. Tako utemeljujemo naš pristop k izvedbi eksperimenta, kjer sta tekmovanji za ljudi in spodbujevalno učenje potekali ločeno in vedno proti računalniškemu igralcu.

7 Literatura

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] C.C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018.

[3] Marc Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47, 07 2012.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. arXiv:1606.01540, 2016.

[5] G. Ciaburro and B. Venkateswaran. *Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles*. Packt Publishing, 2017.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[8] Erik Meijering, Mathews Jacob, Javier Sarria, and Michael Unser. A novel approach to neurite tracing in fluorescence microscopy images. In *Proceedings of the IASTED International Conference on Signal and Image Processing*, volume 5, pages 491–495, 01 2003.

[9] Mayank Mishra. Convolutional neural networks, explained. `https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939`, datum ogleda: 29. 7. 2022.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv:1312.5602, 2013.

[11] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. 10.48550/arXiv.1511.08458, 2015.

[12] AssemblyAI Patrick. Reinforcement learning with (deep) q-learning explained. https://www.youtube.com/watch?v=kEGAMppyWkQ, datum ogleda : 5. 8. 2022.

[13] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.

[14] Kartikeya Rana. Pooling layer — short and simple. https://ai.plainenglish.io/pooling-layer-beginner-to-intermediate-fa0dbdce80eb, datum ogleda: 19. 7. 2020.

[15] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 04:310–316, 05 2020.

[16] Thomas Sinonimi. Diving deeper into reinforcement learning with q-learning. https://www.freecodecamp.org/news/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/, datum ogleda : 4. 8. 2022.

[17] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[18] J. K. Terry, Benjamin Black, and Luis Santos. Multiplayer support for the arcade learning environment. 10.48550/arXiv.2009.09341, 2020.

[19] WatElectronics. What is backpropagation neural network & its working. https://www.watelectronics.com/back-propagation-neural-network/, datum ogleda: 26. 7. 2022.

[20] Mance Harmon Wl, Mance E. Harmon, and Stephanie S. Harmon. Reinforcement learning: A tutorial, 1996. https://www.cs.toronto.edu/~zemel/documents/411/rltutorial.pdf (datum ogleda: 4. 8. 2022).