Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.85
DOI: https://doi.org/10.24867/02BE34Lalic

ANALIZA PERFORMANSI ALGORITAMA UČENJA USLOVLJAVANJEM U OKVIRU STARCRAFT 2 OKRUŽENJA

ANALYZING THE PERFORMANCES OF DEEP REINFORCEMENT LEARNING ALGORITHMS IN THE STARCRAFT 2 ENVIRONMENT

Saša Lalić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj *–Ovaj rad se bavi analizom performansi algoritama učenja uslovljavanjem pri rešavanju problema iz kompjuterske igre Starcraft 2. Algoritmi koji su implementirani i poređeni su A3C i Deep-Q Learning. Za svaki algoritam ispitan je uticaj različitih parametara obuke, kao što su broj preskočenih koraka agenta, i stopa učenja neuronske mreže. Pokazalo se da oba algoritma reaguju isto na promene parametara, i da u problemima koji ne zahtevaju česte akcije da bi dostiglo optimalno rešenje, preskakanje većeg broja akcija ubrzava obuku algoritma, i dovodi do boljeg rešenja u istom vremenskom periodu za obuku. Dok smanjenje stope učenja dovodi do lošijeg rešenja u svim slučajevima. Oba algoritma su postigli rezultate u problemima upravljanja jedinica, ali nisu postigli značajne rezultate u izgradnji baze.*

Abstract *– This paper studies the performance of deep reinforcement learning algorithms in solving a subset of problems in the Starcraft 2 environment. Algorithms that were studied were: A3C and Deep-Q Learning. Each algorithm was tested with a different set of training parameters, such as the number of skipped agent steps, and the neural network learning rate. Both algorithms perform similarly in accordance to parameter changes, when dealing with the problems that do not require a great number of actions to achieve an optimal solution. Consequently parameters values that skip greater number of actions lead to better results given the same training time. Reducing the learning rate leads to a decrease of performances for both algorithms in all of the problems. Both algorithms have achieved satisfiable results in problems that mostly involve management of units. However the results were quite low tasks that included the construction of a base.*

Ključne reči: *Starcraft 2, učenje uslovljavanjem, duboko učenje, A3C, Deep-Q learning*

1. UVOD

Oblast mašinskog učenja se bavi razvijanjem algoritama i tehnika rešavanja problema koji nisu primereni klasičnim tehnikama, i koji rešavaju kompleksne probleme koji su nekad mogli rešiti samo ljudi. Jedna oblast koja se dobro pokazala je učenje uslovljavanjem (*reinforcement learning*).

Reinforcement learning se zasniva na interakciji sa pravim ili simuliranim okruženjem i bavi se algoritmima koji direktno interaguju sa okruženjima.

Reinforcement learning agent interaguje sa svojim okruženjem u diskretnim vremenskim koracima. U svakom trenutku *t* agent dobija opažanje (opis stanja okruženja) nakon čega on bira akciju iz skupa trenutno dostupnih akcija koju prosleđuje okruženju. Okruženje onda prelazi u sledeće stanje. Cilj agenta je da skupi što više nagrade može. Nagrada predstavlja meru uspeha algoritma. Moguće je uvesti kaznu, koja je mera neuspeha agenta. Glavne razlika u odnosu na nadgledano učenje su to što *reinforcement learning* ne zahteva postojanje tačnih parova ulaznih i izlaznih vrednosti, i nije potrebna eksplicitna ispravka neoptimalnih akcija. Umesto toga je fokus na performansama, što uključuje nalaženje ravnoteže između istraživanja (skupljanja novih podataka) i iskorištavanja postojećih podataka. *Reinforcement learning* pruža rešenja u poljima robotike kao i u oblasti igranja video igara.

Ovaj rad predstavlja i poredi rezultate obuke algoritama *reinforcement learning*-a u rešavanju nekoliko problema iz igre *Starcraft 2* [2]. *Starcraft 2* je je vojna naučno fantastična strategija u stvarnom vremenu koju razvija i objavljuje *Blizzard Entertainment*. Korišćeni algoritmi su Q-učenje (*Q-learning*) i Asinhroni prednost akter-kritičar (*Asynchronous Advantage Actor-Critic*, A3C). Oni predstavljaju najpoznatije algoritme u upotrebi za rešavanje problema iz domena kompjuterskih igara. Problemi nad kojima se algoritma obučavanju predstavljaju skupljanje resursa (*CollectMineralShards*), traženje neprijatelja (*FindAndDefeatZerglings*), borba sa neprijateljem (*DefeatRoaches)* i izgradnja baze (*BuildMarines*). Algoritmi su trenirani sa nekoliko različitih skupova parametara da bi se utvrdila podešavanja pod kojima algoritmi postižu najbolje rezultate za dati problem. Najbolji rezultati su postignuti od strane A3C algoritma, dok je najbolji skup parametara za oba algoritma bio povećanje step_mul (vrednost koja određuje koliko koraka agent preskače pri interakciji sa okruženjem) vrednosti što je dovelo do brže obuke u slučaju skupljanja resursa i traženja neprijatelja, a lošije u slučaju borbe. Nijedan algoritam nije uspeo da postigne značajne rezultate u izgradnji baze u vremenu za obuku koje smo mi imali.

Preostali deo rada organizovan je na sledeći način. U narednoj sekciji naveden je uporedni prikaz prethodnih rešenja i vladajućih stavova na temu sentiment analize. Treća sekcija posvećena je definiciji problema i metodologiji za njegovo rešavanje. Četvrta sekcija se bavi

---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kovačević, red.prof.

evaluacijom dobijenih rezultata, a peta sumarizacijom rada i pravcima daljeg razvoja.

2. PRETHODNA REŠENJA

Reinforcement learning primenjen na rešavanje problema u video igrama je tema mnogih radova. Rešavanjem problema *Starcraft 2* se bavilo dosta radova, pokrivajući različite probleme, algoritme, tehnike obuke. U [3] se uvodi SC2LE okruženje, okruženje za *reinforcement learning* bazirano na igri *Starcraft 2*. Pored specifikacije *Starcraft 2* domena, pružaju *Python* bazirani interfejs za komunikaciju sa okruženjem igre, niz mini igara koji se fokusiraju na različite elemente *Starcraft 2 gameplay*. *Replay*[1] podatke pravih ekspertnih igrača kao i bazne rezulutate za neuronske mreže trenirane na osnovu ovih podataka da predvide rezultate igara i akcija igrača. Na kraju ovaj rad predstavlja bazne rezultate *reinforcement learning* agenata nad *Starcraft 2* domenom. Rad [3] predstavlja osnovu od koje naš rad počinje. U njemu su zadati i kreirani radni okviri i biblioteke koje omogućavaju i olakšavaju *reinforcement learning* nad problemima *Starcraft 2*. U [1] se opisuje pysc2 okruženje za učenje i procenjuju delotvornost A3C algoritma u rešavanju nekoliko mini mapa. Fokus rada je poređenje različitih arhitektura neuronskih mreža A3C algoritma. Pored kraće diskusije individalnih problema koji su rešavani, ovaj rad obrađuje i rezultate transfer learning-a (uzimanje težina neuronske mreže nakon obuke nad jednim problemom, i korišćenjem istih za početak obuke mreže nad drugim). Rezultati ovog rada su mnogo pogodniji poređenju sa našim od [3] zato što su za treniranje algoritama autori imali značajno manji broj iteracija.

U [4] je izvršena procena primerenosti *reinforcement learning* algoritama za rešavanje zadatka mikro menadžmenta borbenih jedinica (pozicioniranje i upotreba posebnih sposobnosti jedinica za svaku ponaosob umesto grupno) u komercijalnoj *real time strategy (*RTS) igri *Starcraft: Broodwar* (preteča *Starcraft 2*). Primenjene tehnike su varijacije *Q-learning* (detaljnije u 3.5.1) i Sarsa (λ) algoritama, jednostavne jednokoračne verzije kao i sofisticiranijih verzija koje koriste *eligibility trace*[2] da nadoknade problem odgođenih nagrada[3]. Cilj je dizajn agenta koji može da uči nenadgledan u kompleksnom okruženju, i koji bi eventualno preuzeo zadatke koje je dotad rešavala neadaptivna deterministička igrina umetna inteligencija (*Artificial intelligence*, AI). U [6] se obrađuju moderni i klasični pristupi *reinforcement learning*. Od većeg značaja nama je njihova implementacija A2C (preteča A3C algoritma koja koristi samo jednog agenta za obuku) , opis i diskusija arhitektura mreže i korišćenog algoritma koja upotpunjuje i potvrđuje rezultate iz [3].

3. METODOLOGIJA I ALATI

U ovoj sekciji se nalazi pregled svih alata, okruženja, biblioteka i algoritama korištenih u okviru ovog rada.

3.1. *Starcraft 2*

Starcraft 2 je naučno fantastična, vojna, *real-time* strateška video igra koju razvija i izdaje *Blizzard Entertainment*. Izašla je u Julu 2010 za *Windows* i *Mac OS X* operativne sisteme.. Verzija korišćena za ovaj rad je 4.2.4.

3.2. *PySC2 – StarCraft 2 Learning Environment*

PySC2 [5] je *Python* komponenta *za Starcraft 2 Learning Environment* koju je napravio *DeepMind*. Otkriva *Blizzard Entertainment Starcraft 2 Machine Learning* API kao *Python Reinforcement Learning* okruženje. PySC2 pruža interfejs *reinforcement learning* agentima za interakciju sa *Starcraft 2*, za dobijanje posmatranja (podataka o stanju igre, položaj jedinica, dostupne akcije, itd.) i slanje akcija.

3.3. CUDA

NVIDIA CUDA biblioteka [8] za duboke neuronske mreže je GPU ubrzana biblioteka primitiva za duboke neuronske mreže. Pruža veoma optimizovane implementacije standardnih rutina kao što su konvolucija unapred i unazad, *pooling*, normalizacija i aktivacione funkcije. CUDA ubrzava često korištene radne okvire za duboko učenje uključujući *Caffe2*, MATLAB, *Microsoft Cognitive Toolkit*, *Tensorflow* i *Theano*.

3.4. Tensorflow

Tensorflow [7] je *open source* simbolička matematička biblioteka korištena za programiranje toka podataka i aplikacija mašinskog učenja kao što su neuronske mreže. Pruža API-je za *python*, C++, *Haskel*, *Java*, *Go* i *Rust*. Takođe je moguće koristiti biblioteke kao što su *Keras* koje bi pojednostavile kreiranje algoritama ili dodale dodatni sloj apstrakcije radi lakše promene biblioteka.

3.5. *Reinforcement learning*

Reinforcement learning je oblast mašinskog učenja inspirasana psihologijom ponašanja, koja se bavi načinom na koji softverski agenti treba da preduzimaju akcije u okruženju da bi maksimizirali neki pojam kumulativne nagrade. *Reinforcement learning* se razlikuje od standardnog nadgledanog učenja u tome da nije potrebno da postoje ispravni ulaz/izlaz parovi, i neoptimalne akcije ne moraju biti eksplicitno ispravljene. Fokus je umesto toga na performansama, što zahteva nalaženje ravnoteže između istraživanja (neistraženog prostora) i iskorištavanja (trenutnog znanja) [9]. Agent uči interagujući sa svojim okruženjemu diskretnim vremenskim koracima. U svakom trenutku *t*, agent prima opažanje, koje tipično uključuje nagradu. Onda agent bira akciju iz skupa dostupnih akcija, koju prosleđuje okruženju. Okruženje onda prelazi u sledeće stanje i određuje nagradu asociranu sa prelazom između ovih stanja. Cilj agenta u *reinforcement learning* je da skupi što veću nagradu može. Značajni elementi *reinforcement learning* su upotreba uzoraka zarad optimizacije performansi, i upotreba funkcija aproksimacije da bi se izborili sa velikim okruženjima. Zahvaljujući njima se

[1] Replay je interaktivan snimak igre, koji sadrži podatke o stanju jedinica i drugih elemenata igre, a ne samo sliku i zvuk.

[2] Privremeni zapis uslova koje dovode do nekog događaja, *eligibility trace* obeležava parametre koji su asocirani sa događajem kao podložne promenema tokom učenja. Na ovaj način *eligibility trace* premošćava jaz između događaja i informacija za učenje.

[3] nagrada koje se dešavaju dalje u budućnosti, problem je dati akcijama koje dovode do njih odgovarajuću težinu, da bi one bile zanemarene i ispraćene ako dovode do bolje nagrade od drugih akcija

reinforcement learning može koristiti u situacijama gde model okruženja nije poznat ali postoji analitičko rešenje ili ako postoji samo simulacija modela okruženja, i interakcija je jedini izvor podataka o okruženju.

3.5.1. *Q-Learning*

Q-learning je tehnika *reinforcement learning* korištena u mašinskom učenju. Cilj je naučiti politiku koja govori agentu koju akciju da preduzme pod kojim okolnostima. Ne zahteva model okruženja i može rešiti probleme sa stohastičnim prelazima i nagradama bez izmena. *Q-learning* skladišti svoje podatke u tabele, pristup koji loše funkcioniše sa većim brojem stanja ili akcija. *Q-learning* može biti kombinovan sa aproksimacijom funkcija, što omogućava primenu algoritma na većim problemima čak i ako je prostor stanja kontinualan. Jedno rešenje je koristiti veštačku neuronsku mrežu kao aproksimator funkcije. Algoritam je funkcija koja računa kvalitet kombinacije stanja i akcije. Pre nego što učenje počne *Q* se inicijalizuje na arbitrarnu fiksnu vrednost. Onda u svakom vremenu *t* agent bira akciju, posmatra nagradu, ulazi u sledeće stanje i koriguje *Q*. Jezgro algoritma je iteracija vrednosti, koja koristi težinski prosek (*weighted average*) stare vrednosti i nove informacije. Nagrada date akcije može biti pomnožena faktorom popusta, koji određuje značaj budućih nagrada.

Deep Q-learning je varianta *Q-learning* koji koristi duboku konvolucionu neuronsku mrežu sa slojevima konvolucionih filtera da oponaša efekte receptivnih polja. *Reinforcement learning* je nestabilan kada je nelinearni aproksimator funkcije kao neuronska mreža korišćen da predstavlja *Q*. Ova nestabilnost dolazi iz korelacije koja se nalazi u sekvenci opažanja, činjenica da male izmene *Q* mogu značajno promeniti politiku i distribuciju podataka, i korelacije između *Q* i ciljnih vrednosti. Implementacija korištena u ovom radu je *Deep Q-learning*, koji u svakom koraku za učenje uzima nasumično sortirani podskup (*batch*) svih prelaza stanja/akcija iz prošle iteracije problema.

3.5.2. *Asynchronous advantage actor-critic* (A3C)

A3C algoritam [10] je izdao *Google DeepMind*. Na velikom broju *deep reinforcement learning* zadataka A3C je brži, jednostavniji i bolji nego DQN. Takođe radi na kontinualnim kao i na diskretnim prostorima akcija. Zbog ovoga je postao dominantan algoritam u rešavanju problema sa kompleksnim stanjem i akcionim prostorom.

Actor-critic je kombinacija beneficija iterativnih metoda baziranim na politici i vrednosnih iterativnih metoda kao što *je Q-learning*. A3C će proceniti funkciju vrednosti *V* (koliko je dobro neko stanje) i politiku (skup izlaza verovatnoće akcija). Ove procene vrše potpuno povezani slojevi na vrhu neuronske mreže. Bitno je da agent koristi procenu vrednosti (kritičar) da koriguje politiku (akter) inteligentnije nego tradicionalne metode gradijenta politike[4].

Advantage je način računanja pravila korigovanja politike, uzimajući nagradu nakon uračunatih faktora popusta kao procenu *Q* vrednosti, i oduzimajući od nje rezultat vrednosne funkcije *V*.

Asynchronous se odnosi na mogućnost paralelizacije agenta. A3C algoritam poseduje jednu globalnu neuronsku mrežu, i skup proizvoljne veličine agenata i njihovih okruženja. Svaki agent počinje sa istim parametrima kao i globalna neuronska mreža. On onda interaguje sa svojom kopijom okruženja i skuplja iskustvo. Kada skupi dovoljno iskustva, npr. dođe do kraja okruženja, njegovo iskustvo se koristi za računanje gubitaka vrednosti i politike. Iz njih se izvlače gradijenti za korigovanje parametara mreže i njima se koriguje globalna mreža. Na kraju svake ovakve iteracije agent postavlja vrednosti svoje mreže na vrednosti globalne mreže i nastavlja dalje. Implementacija korištena u ovom radu je ovde opisani A3C algoritam.

4. EKSPERIMENTALNA POSTAVKA

U ovoj sekciji predstavljamo neke od problema u Starcraft 2 i evaluiramo algoritme nad njima. Predstavljamo eksperimentalnu postavku za evaluaciju algoritama. Eksperimentalna postavka se sastoji od problema i primenjenih algoritama. Pod problemima smatraju se nekoliko konkretnih zadataka u igri *Starcraft 2*. Primenjeni algoritmi su Q-learning i A3C, reinforcement learning algoritmi koji su obučavani nad ovim problemima.

4.1. Problemi

Problemi su predstavljeni kao nekoliko malih mapa u *Starcraft 2*. Izabrali smo sledeće probleme da bi smo pokrili specifične probleme iz domena *Starcraft 2*.

CollectMineralShards – agent počinje sa dva marinca (vrsta jedinice u igri) i mora ih selektovati i pomerati po mapi da bi skupio minerale. Svaki mineral vredi jedan poen, raspoređeni su nasumično po mapi, i ako agent uspe da skupi sve minerale pre nego što mu vreme istekne mapa se ponovo puni novim mineralima. Agent ima 2 minuta za jednu iteraciju ove mape.

FindAndDefeatZerglings – agent počinje as 3 marinca i mora istražiti mapu da nađe i pobedi individualne zerglinge. Ovo zahteva pomeranje kamere i efikasno istraživanje. Svaki zergling ubijen vredi 1 poen, svaki marinac izgubljen vredi -1 poen. Agent ima 2.5 minuta za jednu iteraciju ove mape..

DefeatRoaches – agent počinje sa 9 marinaca i mora pobediti 4 roach-a. Svaki put kad pobedi sve roach-eve dobije jos 5 marinaca pojačanja i još 4 roach-a kao protivnike. Svaki roach vredi 10 poena a svaki izgubljeni marinac -1. Agent ima 3 minuta za jednu iteraciju ove mape.

BuildMarines – agent počinje sa ograničenom bazom, i cilj mu je građenje marinaca. Svaki marinac kojeg napravi vredi jedan poen. Dok je prostor akcija njemu ograničen isključivo na akcije potrebne da bi on napravio marince u pitanju je još uvek kompleksan problem. Agent mora skupljati resurse, izgraditi *supply depo*, baraku i onda istrenirati marince tim redosledom. Agent ima 15 minuta za jednu iteraciju ove mape.

4.2 Rezultati

Na osnovu rezultata možemo izvući nekoliko zaključaka. U svim problemima A3C algoritam je značajno bolji od *Q-learning* algoritma. U svim problemima sem u slučaju *DefeatRoaches* problema je treći skup parametara

[4] Gradijent politike je tip tehnike učenja uslovljavanjem koja optimizuje politike koristeći *gradient descent*.

najbolji. Veći broj iteracija dovodi do boljih rezultata algoritma, ako problem ne zahteva brze reakcije da bi bio optimalno rešen. U uslovima gde su resursi za obuku algoritama ograničeni, stoga davanje algoritmu ređe priliku da izdaje akcije ubrzava obuku jer stiže da obradi više iteracija. Drugi skup parametara se pokazao loše u svim slučajevima, iz čega se vidi da su ovi problemi isuviše kompleksni da bi postigli veliki napretke u učenju u početnim iteracijama obuke, na koje drugi skup parametara stavlja fokus. *Q-learning* se pokazao kao lošiji algoritam zbog manje paralelizacije i samim tim manje brzine obuke, ali i zbog toga što *Q-learning* ne računa gubitak politike kao A3C već se samo oslanja na *advantage* pri računanju gubitka funkcije. Naši rezultati su značajno gori od rezultata sličnih istraživanja zbog ograničenja hardvera i vremena, što je smanjilo broj iteracija obuke algoritama, kao količinu različitih skupova parametara obuke koju možemo zadati.

Ljudski rezultati su značajno bolji ne samo od naše implementacije algoritama, već i od implementacija u svim drugim dosadašnjim radovima. Uzrok ovoga je velika kompleksnost *Starcraft 2*, koja još uvek predstavlja izazov sadašnjim *reinforcement learning* algoritmima.

Tabela 1. *Uporedni rezultati evaluacije svih algoritama*

Tip problema/Algoritam	**A3C**	**Deep Q-learning**
CollectMinerals	34.63	**0.63**
CollectMinerals (manji *learning rate)*	28.68	**0**
CollectMinerals (manja učestalost akcija)	36.91	**2.98**
FindAndDefeatZerglings	7.45	**0**
FindAndDefeatZerglings (manji lr)	2.58	**-1.72**
FindAndDefeatZerglings (manja učestalost akcija)	8.44	**6.1**
DefeatZerglings	37.37	**12.23**
DefeatZerglings (manji lr)	9.99	**0**
DefeatZerglings (manja učestalost akcija)	6.99	**4.09**
BuildMarines	0	**0**
BuildMarines (manji lr)	0	**0**
BuildMarines (manja učestalost akcija)	0	**0**

5. ZAKLJUČAK

Problem koji je u ovom radu rešavan je implementacija i analiza performansi A3C i *Q-learning* algoritama u rešavanju problema iz domena *Starcraft 2*.

Ovi algoritmi su implementirani uz oslonac na *Tensorflow* i pysc2 biblioteke, napisane u programskom jeziku *python*. Oba algoritma su implementirana sa sličnom arhitekturom neuronske mreže, sa dva konvoluciona sloja i potpuno povezanim slojem za izlaze neprostorne akcije i vrednosti stanja, i *softmax* slojem za izlaz prostorne akcije. *Q-learning* algoritam računanje *advantage* vrši u izlaznim slojevima neuronske mreže, zbog čega njegovi izlazni slojevi imaju podelu na dva odvojena sloja, dok A3C algoritam *advantage* računa u koraku ažuriranja algoritma, pa je njegova arhitektura jednostavnija. A3C algoritam podržava obuku koristeći više agenata istovremeno, tako što vrši ažuriranje neuronske mreže posle jedne iteracije nad problemom svakog agenta, dok *Q-learning* radi samo sa jednim agentom. *Q-learning* sadrži dve identične neuronske mreže, od kojih jednu koristi samo za generisanje ciljnih vrednosti u obuci, i koju izjednačava sa glavnom neuronskom mrežom svakih 50 koraka obuke da bi postigao stabilniju i konzistentniju obuke.

Svaki algoritam je treniran osam sati u realnom vremenu nad 4 problema: *CollectMinerals*, *DefeatRoaches* i *BuildMarines*, *FindAndDefeatZergling*. Svaki algoritam je takođe treniran sa 3 skupa različitih parametara za obuku za svaki problem. Rezultati za svaki algoritam su dobijeni uzimajući prosečan rezultat 100 iteracija nad problemom po završetku obuke.

Pokazali smo da su svi izabrani problemi sa izuzetkom BuildMarines problemi koje *i Q-learning* i A3C algoritam mogu savladati, sa dovoljno vremena obuke. Naša arhitektura algoritama se loše pokazala na BuildMarines, ali zbog dužine i kompleksnosti mape, teško je utvrditi da li je problem manjak vremena obuke, sa obzirom da su u pysc2 radu tek posle nekoliko miliona koraka obuke uspeli postići neke rezultate, nešto što prevazilazi resurse nama na raspolaganju. Dalji pravci istraživanja uključuju modifikaciju algoritama specifično za svaki problem, isprobavanje većeg broja paramatera obuke, i rešavanje drugih problema u okviru *Starcraft 2*. Mogući pravac takođe predstavlja primena ovih algoritama na druge kompleksne igre, kao što je Dota 2.

6. LITERATURA

[1] A. Basel, P. G. Keerthana "Asynchronous Advantage Actor-Critic Agent for Starcraft II", 22.7.2018,

[2] Starcraft 2 Windows PC version, Blizzard Entertainment, 2010.

[3] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, "StarCraft II: A New Challenge for Reinforcement Learning", 16.08.2017.

[4] S. Wender, I. Watson, "Applying Reinforcement Learning to Small Scale Combat in the Real-Time Strategy Game StarCraft:Broodwar", 2012

[5] https://github.com/deepmind/pysc2

[6] R. Ring, "Replicating DeepMind StarCraft II Reinforcement Learning Benchmark with Actor-Critic Methods", 2018

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015, Software available from tensorflow.org.

[8] NVIDIA cuDNN, https://developer.nvidia.com/cudnn

[9] L. Kaelbling, M. Littman, A. Moore, "Reinforcement Learning: A Survey", Journal of Artificial Intelligence Research. 4: 237–285, Archived from the original on 20.11.2001. (1996).

[10] A. Juliani, "Asynchronous Actor-Critic Agents (A3C)", 17.12.2016.

KRATKA BIOGRAFIJA

Saša Lalić je rođen 12.08.1994. godine u Subotici. Osnovnu školu „Hunjadi janoš" u Čantaviru završio je 2009. godine. Gimnaziju „Svetozar Marković " u Subotici završio je 2013. god. Iste godine upisao se na Fakultet tehničkih nauka, odsek Računarstvo i automatika. Osnovne studije je završio 2017. godine, nakon čega upisuje master akademske studije na Fakultetu tehničkih nauka, smer Računarstvo i automatika. Položio je sve ispite propisane planom i programom.