Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/06BE05Sarenac

PRIMENA UČENJA USLOVLJAVANJEM NA OBUČAVANJE AGENTA ZA AUTONOMNO KRETANJE U OKRUŽENJU *CarRacing-v0*

USING REINFORCEMENT LEARNING TO TRAIN AN AGENT FOR THE *CarRacing-v0* ENVIRONMENT

Novica Šarenac, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U radu je opisan trening i evaluacija agenta za autonomno kretanje u OpenAI Gym okruženju CarRacing-v0. Okruženje predstavlja stazu za vožnju automobila iz ptičije perspektive. Za treniranje agenta korištene su tehnike učenja uslovljavanjem. Poređene su osobine algoritama u pogledu rezultata postignutih u okruženju, dužine treniranja i implementacionih detalja. Implementirani i poređeni algoritmi su: Deep Q-Network (DQN), Advantage Actor Critic (A2C) i Asynchronous Advantage Actor Critic (A3C).*

Ključne reči: *učenje uslovljavanjem, CarRacing-v0, Deep Q-Network, Advantage Actor Critic, Asynchronous Advantage Actor Critic*

Abstract – *This paper presents training and evaluation of the agent for autonomous driving in OpenAI Gym environment CarRacing-v0. Environment is a top-down view of racing track. Agent is trained using reinforcement learning techniques. Algorithms are compared in terms of results achieved in the environment, training time and implementation details. Algorithms that are implemented and evaluated are: Deep Q-Network (DQN), Advantage Actor Critic (A2C) i Asynchronous Advantage Actor Critic (A3C).*

Keywords: *reinforcement learning, CarRacing-v0, Deep Q-Network, Advantage Actor Critic, Asynchronous Advantage Actor Critic*

1. UVOD

U današnje vreme prisutna je velika ekspanzija veštačke inteligencije u svim sferama računarskih nauka. Sve više se teži razvoju softvera koji se na neki način može okarakterisati kao inteligentan. Veliku ulogu u toj ekspanziji ima razvoj hardvera, koji omogućava treniranje složenih modela.

Najčešća upotreba veštačke inteligencije je u detektovanju i prepoznavanju objekata na slikama, razumevanju i klasifikaciji teksta, sistemima za preporučivanje, odgovaranju na pitanja, i slično.

Sve češće veštačka inteligencija se upotrebljava za obučavanje inteligentnih agenata koji fleksibilno i samostalno izvšavaju neke akcije u okruženju.

Tako u auto-industriji mnoge kompanije rade na razvoju autonomnih vozila, u privredi mnogi procesi teže ka automatizaciji, u softveru se razvijaju agenti čiji je cilj ostvarenje što boljih rezultata u simuliranim okruženjima (poput video igara).

U ovom radu opisan je trening i evaluacija agenta za autonomno kretanje u simuliranom okruženju. Okruženje predstavlja vožnju automobila iz ptičije perspektive. Za treniranje agenta korišćena su i upoređena tri algoritma učenja uslovljavanjem: *Deep Q-Network (DQN) [1], Advantage Actor Critic (A2C) i Asynchronous Advantage Actor Critic (A3C) [2].*

2. METODOLOGIJA

Svaki od pristupa za treniranje agenta koristi različite korake preprocesiranja stanja (observacija) koja daje okruženje, kao i različite metode dubokog učenja uslovljavanjem.

2.1. Okruženje *CarRacing-v0*

CarRacing-v0 [3] je okruženje sa *OpenAI Gym* platforme. Okruženje podrazumeva 2D simulaciju upravljanja automobilom iz ptičije perspective.

Stanje koje okruženje vraća prilikom interakcije je slika dimenzija 96x96 piksela, sa tri kanala (RGB).

Nagrade se računaju za svaki frejm (korak u okruženju). Nagrada je -0.1 za svaki frejm i 1000/N za svaku posećenu pločicu na stazi, gde je N ukupan broj pločica na stazi. Kako se staza generiše na slučajan način za svaku igru, ukupan broj pločica na stazi varira između 230 i 320. Epizoda se završava kada se posete sve pločice na stazi ili kada prođe 1000 frejmova.

Jedna od najvažnijih karakteristika ovog okruženja je što ima kontinualne akcije, za razliku od većine okruženja, kao što su *Atari* igre koje imaju diskretne akcije. Svaka akcija sastoji se od tri kontinualne vrednosti, koje odgovaraju: smeru (levo ili desno), ubrzanju i kočenju. Upravljanje smerom je u rangu između -1.0 i 1.0, gde je -1.0 potpuno skretanje levo i 1.0 potpuno skretanje desno. Ubrzavanje i kočenje je u rangu od 0 do 1.

U donjem delu prozora nalaze se grafički predstavljeni indikatori: brzina, *ABS* senzori, pozicija volana i žiroskop. Ove informacije nisu dostupne u brojevnom obliku. Smatra se da je okruženje rešeno ako je srednja vrednost nagrade u 100 uzastopnih pokušaja 900.

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

2.2. Primena DQN metode

Jedna od tri primenjene metode je DQN metoda. Da bi se agent mogao uspešno obučavati potrebno je stanje koje se dobija od okruženja prilagoditi DQN metodi. Takođe je potrebno odabrati optimalnu arhitekturu neuronske mreže, koja je pogodna za rad sa takvim stanjem.

2.2.1 Stanje u DQN metodi

Okruženje *CarRacing-v0* reprezentuje svako stanje sa 96x96x3 *RGB* matricom. Iz razloga kao što su brža konvergencija i trening vrši se niz koraka preprocesiranja originalnog stanja.

Stanje se prvo prebacuje u *grayscale*, čime se dobija matrica dimenzija 96x96. S obzirom na to da se indikatori sa donjeg dela prozora ne mogu iskoristiti u brojevnom obliku, oni se uklanjaju iz stanja. Iseca se prostor dimenzija 12x96 sa donjeg dela svake slike. Takođe se iseca prostor dimenzija 96x6 piksela sa leve i desne strane svake slike. Nakon izvršenih koraka preprocesiranja dobija se slika dimenzija 84x84 piksela.

Takođe se primenjuje tehnika *frame-skipping*. Agent vidi i odabira akciju za svako *k*-to stanje, umesto za svako stanje. Poslednja odabrana akcija se ponavlja za preskočena stanja. Pravljenje koraka u okruženju je računski veoma jeftino i brzo, za razliku od odabiranja akcija, pa ova tehnika omogućava agentu da odigra *k* puta više igara bez velikog povećanja vremena traniranja. Vrednost korištena za parametar *k* je 4.

2.2.2 Arhitektura DQN mreže

Stanje koje se koristi za obučavanje agenta je slika ekrana okruženja, pa je odabrana arhitektura koja predstavlja konvolutivnu mrežu. Arhitektura se sastoji od konvolutivnih, *batch normalization* i potpuno povezanih slojeva. Ulaz u neuronsku mrežu je preprocesirana slika dimenzija 84x84.

Prvi sloj je 2D konvolutivni sloj. Broj kanala na izlazu tog sloja je 16. Koristi se filter dimenzija 5x5 i korak (*stride)* pomeranja filtera 2. Sledeći sloj je *batch normalization* sloj.

Drugi konvolutivni sloj na ulazu očekuje 16 kanala, a broj kanala na izlazu je 32. Dimenzije filtera u ovom sloju su 5x5, a korak pomeranja 2. I nakon ovog sloja vrši se *batch* normalizacija.

Broj kanala na ulazu i izlazu trećeg konvolutivnog sloja je 32. Koristi se filter dimenzija 5x5 sa korakom pomeranja 2. Sledeći sloj je takođe *batch* normalizacija. Izlaz se transformiše u jednodimenzioni vektor, koji je pogodan za potpuno povezane slojeve.

Poslednja dva sloja su potpuno povezani slojevi. Prvi potpuno povezan sloj na ulazu ima dimenziju 1568, a na izlazu 256. Poslednji sloj na ulazu ima dimenziju 256, a na izlazu 4, što se poklapa sa brojem akcija. Izlaz neuronske mreže predstavlja *Q*-vrednosti za svaku akciju.

2.2.3 Implementacija DQN metode

Obučavanje agenta *DQN* metodom koristi okruženje i korake preprocesiranja opisane u sekciji 2.2.1. U implementaciji se koriste dve mreže: *policy* mreža i *target* mreža. *Policy* mreža koristi se za odabiranje akcija, a *target* mreža služi za računanje očekivanih *Q* vrednosti.

U *replay* memoriji se čuvaju parovi: stanje, akcija, nagrada, sledeće stanje i da li je epizoda završena. Kapacitet *replay* memorije koji je korišten je 10 000 primera. Tokom treniranja izvršava se glavna trening petlja. Broj izvršenih koraka u trening petlji je 150 000.

Prvi korak u trening petlji je računanje *Q*-vrednosti za akcije u trenutnom stanju. Za ovo računanje koristi se *policy* mreža. Akcije su diskretizovane, tako da postoje četiri moguće akcije: skretanje levo, skretanje desno, gas i kočnica. Tokom treninga, za odabir akcije se koristi ε-*greedy,* da bi agent mogao istraživati okruženje. Ukoliko je slučajno odabran broj manji od ε agent na slučajan način odabira akciju. U suprotnom uzima se akcija za koju je Q-vrednost maksimalna. Vrednost ε se menja tokom treninga (1).

$$\varepsilon = \varepsilon_{final} + \left(\varepsilon_{start} - \varepsilon_{final}\right) * e^{(-1*step/\varepsilon_{step})} \quad (1)$$

Vrednost ε_{start} je 0.9, ε_{final} je 0.05 i ε_{step} je 30 000. Parametar *step* je korak do koga je trening petlja stigla.

Sledeći korak je izvršavanje akcije u okruženju. Nakon toga se par koji se sastoji od stanja, akcije, novog stanja (dobijenog iz okruženja) i statusa epizode čuva u *replay* memoriji.

Update policy mreže je sledeći korak obučavanja. Prvo se na slučajan način uzima *batch* sakupljenih primera iz *replay* memorije. Veličina batch-a je 32. Računaju se Q-vrednosti za urađene akcije za sva stanja u *batch*-u, koristeći *policy* mrežu. Zatim se računaju očekivane Q-vrednosti za sledeća stanja iz *batch*-a koristeći *target* mrežu. Nakon toga se računa loss i vrši *update policy* mreže.

Update target mreže vrši se povremeno, na svakih 10 000 koraka. *Update* se vrši tako što *target* mreža preuzme parametre *policy* mreže. Algoritam za optimizaciju i *update* neuronske mreže korišten u implementaciji je *RMSprop* [4].

2.3. Primena *actor-critic* metoda

Actor-critic metode koje su implementirane su A3C i A2C. Kao i kod DQN metode, i za *actor-critic* metode je potrebno prilagoditi stanje koje se dobija od okruženja i pronaći optimalnu arhitekturu neuronske mreže. Za obe implementirane metode se koriste isti koraci preprocesiranja stanja, kao i ista arhitektura neuronske mreže.

2.3.1. Stanje u *actor-critic* metodama

Preprocesiranje stanja (96x96x3 *RGB* matrice) koje se dobija od *CarRacing-v0* okruženja sastoji se od niza koraka preprocesiranja.

Prvi korak preprocesiranja je prebacivanje slike u *grayscale,* čime se dobija stanje dimenzija 96x96 piksela.

Sledeći korak preprocesiranja je centriranje piksela oko 0. To se vrši oduzimanjem srednje vrednosti od svakog piksela na slici. Kao i kod DQN metode, indikatori sa

donjeg dela prozora se uklanjaju iz stanja i iseca se prostor dimenzija 12x96 piksela sa donjeg dela svakog stanja. Takođe se iseca prostor dimenzija 96x6 piksela sa leve i desne strane stanja. Nakon preprocesiranja, dimenzija stanja je 84x84 piksela.

Krajnje stanje, koje se koristi u *actor-critic* metodama formira se od nekoliko poslednjih observacija (preprocesiranih 84x84 slika) koje daje okruženje. Observacije se čuvaju i stanje koje je ulaz u neuronsku mrežu za A3C i A2C algoritme je dimenzija 84x84x*N*. *N* je broj observacija koji se čuva. Vrednost *N* koja je korištena je 5.

2.3.2 Arhitektura mreže kod A2C i A3C metoda

Arhitektura neuronske mreže korištene kod A2C i A3C metoda je konvolutivna neuronska mreža. Arhitektura se sastoji od konvolutivnih i potpuno povezanih slojeva. Ulaz u neuronsku mrežu opisan je u sekciji 2.3.1 i ima dimenzije 84x84x5.

Prvi sloj je 2D konvolutivni sloj. Broj kanala na izlazu tog sloja je 16. Koristi se filter dimenzija 8x8 i korak pomeranja filtera 4.

Drugi konvolutivni sloj na ulazu ima 16 kanala, a broj izlaznih kanala je 32. Dimenzije filtera u ovom sloju su 3x3, a korak pomeranja filtera 2.

Izlaz drugog konvolutivnog sloja transformiše se u jednodimenzioni vektor, koji je pogodan za potpuno povezane slojeve. Sledeći sloj je potpuno povezan i na ulazu ima dimenziju 2592, a na izlazu 256.

Neuronska mreža ima dva izlaza. Jedan predstavlja funkciju vrednosti, a drugi politiku, Funkcija vrednosti i politika se razlikuju samo u poslednjem sloju.

Ulazna dimenzija za poslednji sloj funkcije vrednosti je 256, a izlazna je 1. Izlaz iz tog sloja je vrednost trenutnog stanja. Isto kao kod funkcije vrednosti, i kod poslednjeg sloja politike ulazna dimenzija je 256. Izlazna dimenzija politike je 4 i poklapa se sa brojem akcija.

2.3.4 Implementacija A3C metode

A3C metoda koristi okruženje i korake preprocesiranja opisane u sekciji 2.3.1. Kako je A3C metoda zasnovana na multiprocesingu prvi korak u implementaciji je da se inicijalizuje više procesa, od kojih svaki ima svoju trening petlju. Broj procesa koji je korišten je 8. Postoji jedan globalni model, čiji *update* vrše svi procesi.

Svaki proces ima lokalnu instancu modela, instancu okruženja, kao i referencu ka globalnom modelu. Inicijalizuje se i memorija koja čuva parove: korak, vrednost stanja, nagrada, logaritam verovatnoće akcije, entropija i da li je epizoda završena. Podaci sačuvani u toj memoriji koriste se za jedan *update* mreže, pa je kapacitet memorije jednak broju koraka koji su potrebni za jedan *update*. Broj koraka korišten u implementaciji je 40. Takođe se izvršava glavna trening petlja. Broj izvršenih koraka u trening petlji u jednom procesu je 150 000.

Prvi korak u trening petlji je *update* lokalne mreže i preuzimanje parametara globalne mreže. Sledeći korak je petlja koja izvršava broj akcija potreban za jedan *update* globalne mreže. Prvo se računaju verovatnoće akcija, logaritmi tih verovatnoća i vrednost trenutnog stanja. Zatim se, koristeći verovatnoće, bira akcija koja će se izvršiti u okruženju. Akcije su kontinualne i dobijaju se tako što se prvo odabere akcija koja ima maksimalnu verovatnoću. Moguće akcije su skretanje levo, skretanje desno, gas i kočnica. Zatim se taj vektor pomnoži sa verovatnoćom i ta akcija se iskoristi kao akcija u okruženju. Nakon odabira akcije računa se entropija, koja podstiče istraživanje agenta.

Sledeći korak u trening petlji je izvršavanje akcije u okruženju, čime se dobija novo stanje i nagrada. Korak, vrednost stanja, nagrada, logaritam verovatnoće akcije, entropija i da li je epizoda završena čuvaju se u memoriji.

Kada se izvrši potreban broj koraka, računa se *loss* i vrši *update* globalne mreže. Prvo se računaju kumulativne nagrade. Zatim, koristeći kumulativne nagrade i vrednosti stanja računa se *advantage* i *loss* za funkciju vrednosti.

Sledeći korak je računanje *loss*-a za politiku, koji koristi izračunati *advantage* i logaritme verovatnoća akcija iz memorije. Koristeći *loss* funkciju vrednosti, *loss* funckiju politike i entropiju računa se krajnja vrednost *loss*-a. Vrednost koeficijenta entropije korištena u implementaciji je 0,01.

Kada se izračuna *loss* vrši se *update* globalnog modela. Algoritam za optimizaciju i update neuronske mreže korišten u implementaciji je Adam [5].

2.3.5 Implementacija A2C metode

A2C metoda koristi okruženje i korake preprocesiranja opisane u sekciji 2.3.1. I A2C metoda zasnovana je na paralelnim agentima, koji istražuju više okruženja. Agenti su sinhroni i koriste isti globalni model, čiji update se vrši tek kada svi agenti izvrše potreban broj koraka. Agenti imaju zajedničku trening petlju.

Inicijalizuju se procesi od kojih svaki ima po jednu instancu okruženja. Broj korištenih procesa je 8. Kada se uradi *reset* ili izvrši korak u okruženjima, čeka se da svi procesi urade *reset* ili korak. Takođe, kao i kod A3C metode, inicijalizuje se memorija, u kojoj se čuvaju parovi: korak, vrednosti stanja, nagrade, logaritmi verovatnoće akcije, entropije i da li su epizode završene. Podaci sačuvani u memoriji služe za *update* neuronske mreže, pa je kapacitet jednak broju koraka koji se izvrše za jedan *update* po procesu. Broj koraka korišten u implementaciji je 5. Broj koraka koji se izvršava u glavnoj trening petlji je 125 000.

Prvi korak u trening petlji je izvršavanje akcija potrebnih za jedan *update* mreže. Izvršava se 5 akcija po procesu. Prvo se računaju verovatnoće akcija, logaritmi tih verovatnoća i vrednost trenitnoh stanja. Od dobijenih verovatnoće bira se akcija koja će se izvršiti u okruženju. Akcije su, kao i kod A3C metode kontinualne i biraju se na isti način.

Nakon odabira akcija računa se entropija. Zatim se odabrane akcije izvršavaju u svim okruženjima u isto vreme. Nakon izvršavanja, od svakog procesa se dobija novo stanje i nagrada.

Koraci, vrednosti stanja, nagrade, logaritmi verovatnoća akcija, entropije i da li su epizode završene čuvaju se u memoriji. Kada se izvrši potreban broj koraka, vrši se *update* mreže. Prvo se računaju kumulativne nagrade. Pomoću njih i vrednosti stanja se računa *advantage*, a zatim i *loss* za funkciju vrednosti. Za računanje *loss*-a za politiku koriste se izračunati *advantage* i logaritmi verovatnoća akcija, koji su sačuvani u memoriji.

Krajnji *loss* se računa pomoću *loss*-a funkcije vrednosti, *loss*-a politike i entropije. Koeficijent entropije koji je korišten je 0,1. Nakon toga vrši se *update* modela. Algoritam za optimizaciju i update mreže korišten u implementaciji je Adam.

3. REZULTATI

Okruženje *CarRacing-v0* smatra se rešenim ukoliko agent ostvari prosečnu vrednost nagrade veću od 900 u 100 uzastopnih pokušaja. Ti rezultati su daleko iznad rezultata koje može ostvariti čovek. Mera tačnosti korištena za evaluaciju sva tri modela je prosečna ostvarena nagrada u 100 uzastopnih epizoda.

Kod DQN metode broj koraka prilikom treninga koji se pokazao kao optimalan je 150 000. Agent i sa manjim brojem koraka (oko 100 000) obučavanja nauči da se kreće unapred, ali ne može da savlada krivine. Tek sa većim brojem koraka agent uči da skreće u krivinama. DQN agent može savladati čak i velike krivine, sa kojima ostali modeli imaju poteškoća. Prosečna osvojena nagrada u 100 uzastopnih epizoda ovog modela je 178.

Optimalan broj koraka prilikom treninga A3C agenta je 150 000. Agent jako brzo (za manje od 40 000 koraka) nauči da prati stazu i kreće se, međutim, kao i DQN agentu, potreban mu je veći broj koraka obučavanja da bi naučio da skreće u kriviama. U 100 uzastopnih epizoda A3C agent ostvaruje prosečnu nagradu 241.

Kod A2C metode optimalan broj koraka prilikom treninga je 125 000. Slično A3C agentu, vrlo brzo nauči da se kreće unapred i prati stazu, ali tek sa velikim brojem koraka obučavanja počinje pravilno skretati u krivinama. Prosečna osvojena nagrada u 100 uzastopnih epizoda ovog modela je 263.

Iz tabele 1. vidi se da A2C metoda ostvaruje najveću prosečnu nagradu u 100 uzastopnih epizoda.

Broj koraka treniranja	Metoda	Prosečna osvojena nagrada u 100 epizoda
Slučajne akcije	-	-54
150 000	DQN	178
150 000	A3C	241
125 000	A2C	263

Tabela 1: Poređenje rezaltata svih implementiranih metoda

4. ZAKLJUČAK

U ovom radu opisano je obučavanje agenta za autonomno kretanje u simuliranom okruženju *CarRacing-v0*. Agent je obučavan koristeći metode učenja uslovljavanjem. Implementirane metode su: *Deep Q-Network* (DQN), *Advantage Actor Critic* (A2C) i *Asynchronous Advantage Actor Critic* (A3C).

Na osnovu rezultata može se zaključiti da agenti trenirani metodama učenja uslovljavanjem mogu naučiti da se snalaze u okruženjima poput *CarRacing-v0*. Metode korištene u radu su se pokazale kao veoma uspešne u učenju direknto iz vizuelnih podataka. Takođe, sve metode su veoma osetljive na promene hiperparametara. Performanse svih modela u velikoj meri zavise od načina preprocesiranja slike i formiranja stanja.

Actor-critic metodama potrebno je mnogo manje vremena da počnu da konvergiraju i uče. Kod njih se učenje može primetiti već nakon 40 000 koraka, gde agent počinje da prati stazu, ali ima probleme prilikom skretanja. Kod DQN metode učenje se može primetiti nešto kasnije, nakon 100 000 koraka obučavanja. Takođe vreme potrebno za trening *actor-critic* metoda je značajno manje nego kod DQN metode, što je rezultat korištenja multiprocesinga.

Metode A2C i A3C imaju slične karakteristike upravljanja automobilom. Agenti obučavani korištenjem A2C i A3C metoda imaju karakteristike brze vožnje i na taj način pokušavaju osvojiti veće nagrade. Ti agenti imaju pravilno skretanje u većini krivina. Agent obučavan korištenjem DQN metode ima karakteristike sporije vožnje, ali pravilnog skretanja u krivinama i na taj način pokušava osvojiti što veću nagradu. Oba načina vožnje imaju prednosti i mane. Problem koji se pojavljuje kod agenata obučavanih A2C i A3C metodama je skretanje u velikim krivinama, dok problem agenta obučenog DQN metodom je povratak na stazu nakon silaska sa staze.

5. LITERATURA

[1] V. Mnih, K. Kavukcouoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, D. Hassabis „*Human-level control through deep reinforcement learning*“

[2] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Harley, T.P. Lillicrap, D. Silver, K. Kavukcouglu „*Asynchronous Methods for Deep Reinforcement Learning*“

[3] https://gym.openai.com/envs/CarRacing-v0/ [pristupljeno 7.9.2019.]

[4] S. Ruder „*An overview of gradient descent optimization algorithms*“

[5] D. P. Kingma, J. L. Ba „*Adam: A Method for Stohastic Optimization*“

Kratka biografija:

Novica Šarenac rođen je u Nevesinju 1994. godine. Osnovne akademske studije iz oblasti Računarstvo i automatika, završio je na Fakultetu tehničkih nauka, 2017. godine., kada upisuje i master akademske studije iz iste oblasti. Položio je sve ispite predložene planom i programom.