Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/10BE20Orcic

UPOTREBA MAŠINSKOG UČENJA ZA OBUČAVANJE ROBOTA ZA ŠPRICANJE JAGODA U UNITY 3D OKRUŽENJU

USE OF MACHINE LEARNING FOR TRAINING A STRAWBERRY SPRAYING ROBOT IN A UNITY 3D ENVIRONMENT

Timotej Orčić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Cilj ovog istraživanja jeste prezentovanje idejnog rešenja rapidnog razvoja veštačko-inteligentnih agenata putem simuliranja u Unity grafičkom okruženju. Oblast primene konkretno razvijanog agenta jeste poljoprivreda. Istreniran je model za detekciju objekata sa slike, kao i Reinforcement Learning model za ciljano kretanje agenta u okruženju. Podaci i okruženje su samostalno kreirani. Svi upotrebljeni modeli i algoritmi su evaluirani.*

Ključne reči: *veštačka inteligencija, veštačke neuronske mreže, prepoznavanje objekata sa slike, Reinforcement Learning, Unity 3D, Unity ML Agents*

Abstract – *The aim of this research is to present a conceptual solution for rapid development of artificially-intelligent agents through simulation in a Unity graphical environment. The area of application of the specifically developed agent is agriculture. An object detection model was trained, as well as a Reinforcement Learning model for targeted agent movement in the environment. Data and the environment are self-created. All models and algorithms used were evaluated.*

Keywords: *artificial intelligence, artificial neural networks, object detection, Reinforcement Learning, Unity 3D, Unity ML Agents*

1. UVOD

Naučno-tehnološka ekspanzija u prethodnim godinama i decenijama otvorila je prostor za automatizaciju velikog broja ljudskih delatnosti, pa tako i poljoprivrednih. San o autonomnim poljoprivrednim sistemima, koji zahtevaju minimalnu ljudsku interakciju sve nam je bliži zahvaljujući enormnom razvoju robotike i kompjuterske inteligencije.

Tema ovog rada je simulacija robota koji šprica jagode (konkretno zrele) obučenog putem *Reinforcement Learning*-a, a data simulacija se odvija koristeći *Unity 3D* grafički *engine* [1]. *Reinforcement Learning* je način obučavanja veštačke inteligencije da izvršava određeni zadatak putem nagrađivanja uspešnih (željenih) akcija i kažnjavanjem neuspešnih, tj. neželjenih akcija veštačko-inteligentnog agenta.

NAPOMENA:

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

Na taj način agent, u našem slučaju robot, može relativno brzo da savlada zadate izazove i "nauči" da izvršava željeni zadatak. Simulacija u grafičkom *engine*-u veoma je pogodna jer isključuje kreiranje stvarnog robota i sve izazove datog postupka, a takođe donosi ogromnu uštedu na vremenu obučavanja veštačke inteligencije jer omogućava paralelizovano treniranje. Odabran je *Unity 3D engine* jer je izuzetno pogodan za brz razvoj kvalitetnih 3D okruženja, a uz to poseduje eksterni skup alata - *Unity ML Agents Toolkit* [2] koji je kreiran baš za namene obučavanja veštačko-inteligentnih agenata u 3D grafičkom svetu.

Kreirana je 3D scena u *Unity*-ju sa tri reda jagoda kroz koje će robot kasnije prolaziti. Nakon toga je napravljen skup od 64 slike datih jagoda koji je podeljen na trening (58 slika) i test skup (8 slika). Jagode na datim slikama su zatim anotirane koristeći *labelImg* kompjuterski program, te je na kraju anotirano ukupno 264 jagode, od kojih je 232 pripadalo trening skupu a 32 test skupu. Zatim je, pomoću datog anotiranog skupa odrađen *fine-tuning* nad *ssd_mobilenet_v1_coco* konvolutivnoj neuronskoj mreži koristeći *Google*-ov *Object Detection API* [3], gde je uvedena klasa *strawberry* (tj. jagoda), kako bi jagode na našoj sceni mogle biti prepoznate od strane robota. Trening je pokrenut na *Google Colab* [4] platformi radi brzine (*GPU*) i trajao je 200,000 koraka. Robotu su postavljene dve kamere, jedna sa desne i jedna sa leve strane, čije su slike direktan ulaz u našu *fine-tune*-ovanu neuronsku mrežu, koje vraćaju poziciju detektovanih jagoda sa datih slika (ukoliko postoje), a nad datim izlazom se još sitnim matematičkim transformacijama zaključuje da li robot treba da pali prskalicu sa te strane ili ne, te mu se to prosleđuje kao krajnja informacija (kroz *Unity* okruženje). Kretanje robota je sprovedeno kroz sistem odvojen od prskanja, gde je u *Unity*-ju, koristeći pogodnosti *ML Agents* paketa isprogramirana cela *RL* logika našeg agenta uz pomoć tri *C#* programske skripte. Nakon još nekih sitnih podešavanja datog *Unity* dela celo okruženje je eksportovano kao aplikacija. Tako eksportovano okruženje učitano je unutar *python* skripte, unutar koje je vršeno obučavanje veštačko-inteligentnog *RL* modela za upravljanje kretanja robota. Obučavanje datog modela vršeno je korištenjem *Asynchronous Advantage Actor Critic RL* algoritma [5], koji omogućava treniranje nad više instanci datog 3D okruženja. Dato obučavanje vršeno je na privatnom računaru (*CPU* sa 4 jezgra) te je učeno na 4 instance okruženja u paraleli. Nakon desetak dana (oko 250 sati) algoritam je načinio 1500 koraka i pokazao dobre rezultate - konkretno agent je naučio da prati prvi red jagoda, da napravi "polukružno skretanje" i da se uspešno kreće između prvog i drugog

reda jagoda, te je obučavanje obustavljeno a model sačuvan kao *Tensorflow* [6] zamrznuti graf (eng. *frozen graph*).
Ovo rešenje može predstavljati osnovu za razvoj stvarnog poljoprivrednog robota koji šprica različite tipove voća ili povrća određenim supstancama, te umnogome olakšati dati deo procesa uzgajanja, a uz dodatni trud može predstavljati osnov za razvoj raznih drugih tipova poljoprivrednih robota, posebno koristeći modul za kretanje.

2. METODOLOGIJA

U ovom poglavlju prezentovane su primenjene metodologije za razvoj pomenutog veštačko-inteligentnog agenta. Dato poglavlje podeljeno je u nekoliko faza, koje prate sam tok razvoja projekta.

2.1. Kreiranje Unity ML okruženja

S obzirom da u *RL* okruženje predstavlja okosnicu dešavanja, na početku je napravljeno *Unity* okruženje, koje sadrži jednu scenu i u koje je uključen *ML Agents* paket. Na datu scenu je ubačen teren (podloga sa teksturom zemlje), osvetljenje i 3D model našeg robota (*RL* agenta). Datom robotu je dodato sve što je neophodno za njegovo kretanje po sceni i njegovu "fiziku".
S obirom da se projekat sastoji iz dva dela (prepoznavanje objekata i *RL*) na taj način je razvijana i *Unity* scena, te je na početku ručno postavljeno par redova 3D modela jagoda i jedna kamera sa desne strane robota, koja je korištena za pravljenje skupa slika opisanog u sledećem delu poglavlja.

2.2. Kreiranje i anotiranje skupa slika jagoda

Napravljeno je 64 slike ručno postavljenih jagoda u našoj sceni, koje će kasnije biti iskorištene za obučavanje modela za detekciju objekata. Treba napomenuti da su date slike jagoda u punoj rezoluciji, a da se robotska kamera služi smanjenom rezolucijom (300X300) radi brzine upotrebe algoritma za detekciju objekata (u realnom vremenu). Anotiranje jagoda na slikama obavljeno je korištenjem *labelImg* alata i anotirano je ukupno 264 jagode (ploda). Nakon toga, dati skup podataka je podeljen na trening (58 slika i 232 jagode) i test skup (8 slika i 32 jagode) i od datih slika i anotacija kreirana su 2 *csv* dokumenta (*training.csv* i *test.csv*) korištenjem male *python* skripte. Zatim su, od datih csv dokumenata kreirana 2 *tfrecord* dokumenta (*train.record* i *test.record*) korištenjem druge *python* skripte, te je ovime završen postupak kreiranja skupa podataka za obučavanje modela detekcije objekata.

2.3. Finetuning object detection modela

Da bismo na najbrži mogući način mogli da naučimo neku mrežu da prepoznaje jagode iz našeg skupa podataka, poslužili smo se *finetuning*-om [7]. Za dati zadatak smo odabrali *ssd_mobilenet_v1_coco* neuronsku mrežu koja je prethodno istrenirana na *coco* skupu podataka. *Coco* [8] skup podataka sadrži 80 klasa, a takođe i nekoliko vrsta voća, što pogoduje našem zadatku. *Finetuning* postupak koji smo uradili jeste obučavanje date mreže da detektuje novu klasu *strawberry* i na taj način omogući prepoznavanje jagoda iz našeg 3D okruženja. Za izvršavanje datog zadatka poslužili smo se *Google*-ovim *API*-jem za treniranje i rad sa *object detection* modelima - *Object Detection API*, koji je razvijen nad *Tensorflow* bibliotekom. Da bismo dobili na brzini i računarskoj snazi (*GPU*), dati zadatak smo izvršili na *Google Colab* platformi, pokrećući par jednostavnih skripti. Da bi to bilo omogućeno bilo je neophodno kopirati *Object Detection API* paket na *Google Drive*, te zatim kroz jednu jednostavnu *Colab* skriptu aktivirati dati paket. Zatim smo na dati *Google Drive* preuzeli istrenirani *SSD* model i raspakovali ga, putem druge jednostavne *Colab* skripte. Na kraju smo, pokretanjem poslednje *Colab* skripte inicirali *finetuning* datog modela za 200,000 koraka koje je teklo veoma brzo (oko 5 sati) jer je odabrani model veoma "lagan" a naš skup podataka nije velik. Na kraju je bilo potrebno eksportovati graf zaključivanja (eng. *inference graph*) kako bi model mogao da se stavi u upotrebu.

2.4 Nameštanje Unity ML okruženja za RL

U prethodnom delu datog poglavlja dali smo kratak uvod u izgled *Unity* okruženja koje smo razvijali, tj. prvog dela razvoja datog okruženja potrebnog za prvi deo projekta. Nakon realizacije prvog dela projekta preostao nam je komplikovaniji deo, *Reinforcement Learning*, tj. obučavanje našeg robota da se kreće u datom okruženju (uz određena pravila) i da postigne određeni cilj.
Za početak je bilo bitno podesiti *Unity* okruženje da bude pogodno za *RL*. To je postignuto korištenjem *Unity ML Agents* paketa, koji pruža svu potrebnu podršku za realizaciju *RL* okruženja i programiranje istih putem *C#* skripti (koje su standardni deo *Unity engine*-a). S obzirom da ključni delovi *Unity ML Agents* implementacije predstavljaju 3 komponente - *Academy*, *Agent* i *Brain*, rasparčaćemo razvoj ovog dela projekta na te 3 komponente. Uz te 3 osnovne komponente, bitno je napomenuti na koji način je razvijeno samo okruženje, tj. na koji način su programski postavljeni određeni elementi na scenu datog okruženja.

2.4.1 RobotArea (oblast u kojoj će se robot kretati)

U prethodnom delu o *Unity* implementaciji spomenuto je da je na scenu našeg okruženja postavljen teren, osvetljenje i 3D model robota. Kako bi postavka terena mogla više puta da se iskoristi dati teren i robot su spakovani u jedan *Unity*-jev *GameObject*, tj. u zajedničkog roditelja, kako bi mogli zajedno da se koriste i pomeraju, te je od datog *GameObject*-a napravljen takozvani *Prefab*. Osim navedenog, napisana je jedna *C#* skripta (*RobotArea.cs*) u kojoj je implementirana sva programska logika potrebna za generisanje određenih objekata na sceni i koja će kasnije poslužiti ostalim delovima *ML Agents* arhitekture da programski pristupaju datim generisanim objektima.

2.4.2 RobotAcademy (Academy - glavna ML Agents komponenta)

Napravljen je *Prefab* za akademiju našeg projekta pomoću koje se kontrolišu osnovna podešavanja našeg okruženja i za koju se vezuju *Brain* elementi *Unity ML*-a. Osim toga, napisana je i kratka *C#* skripta koja kontroliše šta se dešava prilikom resetovanja akademije, a u našem slučaju to će samo inicirati resetovanje *RobotArea* instanci koje postoje u našem okruženju (konkretno jedna).

2.4.3 RobotAgent (RL Agent)

Napravljen je *Prefab* za našeg agenta koji sadrži 3D objekat robota i kamere vezane za njega. Dati *Prefab* ima dodat *rigidbody* element koji mu dodaje potrebnu "fiziku" i naravno *C#* skriptu kojom se kontroliše njegova logika

kretanja, a u koju je ugrađena sva *RL* logika pomoću koje će dati agent naučiti da se pravilno kreće u datom okruženju. Na datu *C#* skriptu se kroz *Unity editor* vezuju konkretne instance kamera, a i *Brain* objekat, koji će u stvari upravljati datim agentom.
Akcije koje naš *RL* agent može da preuzme prilikom kretanja su sledeće:

1) Ne radi ništa (ne preduzimaj nikakvu akciju)
2) Skreći levo
3) Skreći desno

Nagrade koje okruženje vraća agentu u odnosu na njegove akcije su sledeće:

1) Kreće se po dobrom putu spram jagoda **+0.15**
2) Kreće se previše blizu/daleko od jagoda **-0.2**
3) Kreće se po dobrom putu spram jagoda u ćošku (prilikom skretanja) **+0.3**
4) Kreće se previše daleko od jagoda u ćošku **-0.4**
5) Kreće se previše blizu jagoda u ćošku **-0.2**
6) Gazi preko jagoda **-5**
7) Ušao je u pogrešan region **-50**
8) Otišao je predaleko od područja s jagodama **-50**
9) Isteklo mu je zadato vreme **-50**
10) Stigao je na cilj **+100**

Dato *RL* okruženje se resetuje u sledećim slučajevima:

1) Agent je ušao u pogrešan region
2) Istekao je zadati vremenski period
3) Agent je otišao predaleko od polja jagoda
4) Agent je stigao na cilj (u toku zadatog vremena)

2.4.4 Brain

Brain (mozak) predstavlja jedinicu kojom se upravlja agentima na koje je povezan. U našem projektu implementirane su 2 vrste mozga za našeg agenta: *RobotPlayeBrain* i *RobotLearningBrain*. Prvi mozak je služio da se postave i testiraju pravila u okruženju i testiraju agentske akcije uz pomoć tastera na tastaturi (poput igrice), dok je drugi mozak kreiran da vrši komunikaciju između *Unity* okruženja i *python RL* koda kroz koji će se naš agent obučavati da uspešno reši zadati zadatak.

Dati agentski mozak nema nikakve vektorske obzervacije, već ima nekoliko **vizuelnih obzervacija**, konkretno, prednju kameru dimenzija 160x120 piksela, crno-belu, koja će se slati na ulaz *python RL* modela i dve kamere sa strane dimenzija 300x300 piksela u boji, pomoću kojih će se aktivirati prskalice prilikom detekcije jagoda. Kao odgovor na dati ulaz prednje kamere, *python RL* model će vraćati vektor dimenzija 1x1 (sa 3 opcije), koji će predstavljati koja je sledeća **akcija** koju agent treba da izvrši u *Unity* okruženju. Dakle, isto ono što smo pomoću *PlayerBrain*-a radili tastaturom, sada će *RL* model učiti da radi sam.

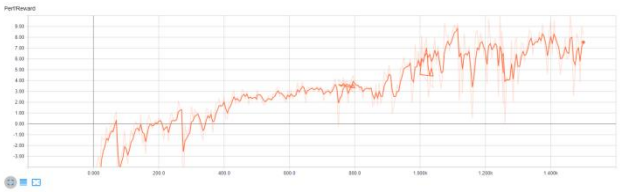
Na kraju je bitno napomenuti da je od datog *Unity* okruženja bilo potrebno napraviti (eng. *build*) program koji može da se izvršava na određenom operativnom sistemu (.exe, .x86_64, ...) kako bi dato okruženje moglo da bude pokrenuto putem *python* koda, te da se izvrši prethodno pomenuto obučavanje *RL* modela.

2.5 Obučavanje RL modela - kretanje robota

Naš agent je "učio" da se kreće koristeći jednu *python* skriptu (*A3C_CPU.py*) u kojoj je kreirana globalna mreža (model) i 4 radnika (eng. *worker*) pomoću kojih se instanciralo naše *Unity* okruženje u 4 paralelne instance (broj *CPU* jezgara privatnog računara).

Za svakog od instanciranog radnika kreirana je lokalna mreža nad kojom se agent obučavao i slao naučene težine u globalnu mrežu, te je tako postignuto deljenje naučenog znanja. Obučavanje je vršeno 1500 koraka i trajalo je oko 10 dana, za kojih je naš agent naučio da solidno prati liniju jagoda i da u većini slučajeva napravi "polukružno skretanje" i isprati drugi red jagoda do kraja.

Na slici 1 prikazan je graf napretka vrednosti nagrade (eng. *reward*) koje je jedan od radnika ostvario u toku obučavanja.



Slika 1. *Graf napretka vrednosti nagrade A3C radnika*

Nakon obučavanja bilo je potrebno napraviti "zamrznuti graf" od obučenog modela, a to je postignuto učitavanjem celokupnog modela (globalni i 4 lokalna), te izdvajanjem i čuvanjem samo globalnog modela putem *tensorflow variable scope*-a, i na kraju "zamrzavanjem" tako izdvojenog globalnog modela, nakon ponovnog učitavanja.

2.6 Povezivanje RL i OD modela u konačan sistem

Nakon obučavanja agenta da se kreće putem *RL* metode, ostalo je samo da se poveže *OD* model za detekciju jagoda sa agentom i na taj način napravi podloga za aktiviranje prskalica sa leve i desne strane robota. U *Unity*-ju su na našeg agenta ugrađene prskalice putem *ParticleSystem GameObject*-a, pomoću kojih je vizualizovana animacija prskanja jagoda. Kao što smo već videli u prethodnom delu datog poglavlja, na naš *LearningBrain* osim prednje kamere robota spojene su i leva i desna kamera.

Kao odgovor na ulaze sa datih kamera *python* skripta za simuliranje našeg agenta vraćaće vektor dimenzija 1x1 (sa 2 opcije) koji će signalizirati da li treba paliti ili gasiti povezanu prskalicu. Data logika je programski povezana kroz *RobotAgent.cs* skriptu.

Krajnje zaključivanje i simulacija sprovedeni su u python skripti *Full_Inference.py*, gde su učitana oba zamrznuta *tensorflow* grafa (*RL* i *OD*), te se za svaki frejm iz datog *Unity* okruženja (prednje kamere) izvršavala *RL* predikcija za upravljanje kretanja robota, a za svaki šesti frejm pozivala detekcija jagoda za levu i desnu kameru, gde se proveravalo da li su detektovane jagode u prvoj trećini ulazne slike i da li su detektovane sa preciznošću većom od 60%.

Ukoliko je dati kriterijum ispunjen, vraćen je signal *Unity* okruženju da se upali prskalica sa odgovarajuće strane.

3. EKSPERIMENTALNA EVALUACIJA I REZULTATI

Izvršena je eksperimentalna evaluacija radi određivanja performansi korištenih algoritama za detekciju jagoda i kretanje robota u prostoru. Performanse algoritma za detekciju jagoda evaluirane su upotrebom test skupa, dok su performanse *A3C* algoritma evaluirane empirijski, tj. posmatranjem ponašanja obučenog *RL* modela. Metrike performansi algoritama za detekciju jagoda koje su korišćene u ovom radu su preciznost, odziv i F_1-mera.

U tabeli 1 prikazani su rezultati *SSD Object Detection* modela pomoću kojeg smo detektovali jagode za različite *IoU* i različite veličine oblasti detekcije.

Tabela 1. *Rezultati OD modela*

IoU/oblasti	Preciznost	Odziv	F1-mera
0.50:0.95/sve	0.745	0.200	0.315
0.50/sve	1.000	0.775	0.873
0.75/sve	1.000	0.775	0.873
0.50:0.95/male	-1.000	-1.000	-1.000
0.50:0.95/srednje	0.722	0.767	0.744
0.50:0.95/velike	0.800	0.800	0.800

Iz tabele se vidi da jedino sitne jagode ne bivaju detektovane, tj. sve metrike za male oblasti imaju vrednost -1.000, dok za srednje i velike oblasti detekcija biva poprilično uspešna.

Evaluacija *A3C RL* algoritma izvršena je empirijski, posmatranjem kretanja koje je robot naučio da savlada. Kao što je pomenuto u prethodnim poglavljima, za 1500 koraka naš robot je naučio da prati prvi red jagoda, da napravi "polukružno skretanje" i da se uspešno kreće između prvog i drugog reda jagoda.

4. ZAKLJUČAK

U ovom radu predstavljeno je jedno rešenje za razvoj veštačko-inteligentnog robota za prskanje jagoda, koje je postignuto simuliranjem u *Unity 3D engine*-u. Kreirana je 3D scena u datom *engine*-u u koju su smešteni redovi jagoda kroz koje će robot kasnije prolaziti, te su napravljene 64 slike datih jagoda koje smo podelili na trening (58 slika, 232 jagode) i test skup (8 slika, 32 jagode). Uz pomoć datog skupa podataka odrađen je *fine-tuning ssd_mobilenet_v1_coco* konvolutivne neuronske mreže, gde je uvedena klasa *strawberry* (tj. jagoda). Dati *OD* model je kasnije iskorišten za prepoznavanje zrelih jagoda sa ulaza leve i desne kamere našeg robota, na osnovu koga reaguje sistem za aktivaciju prskalica (leve i desne). Kretanje robota obučeno je korištenjem *A3C RL* algoritma, uz pomoć pogodnosti koje nudi *Unity ML Agents* paket i njegova mogućnost povezivanja sa *python*-om.

OD model je pokazao zadovoljavajuće rezultate, s obzirom na ne preterano velik skup podataka. Evaluacija datog modela vršena je nad test skupom, a korištene su sledeće metrike: preciznost, odziv i F1-mera. Najveća preciznost od 1.000 postignuta je za *IoU* 0.5 i 0.75 nad svim oblastima, najveći odziv od 0.800 za *IoU* 0.50:0.95 nad velikim oblastima, a najveća F1 mera od 0.873 vezana je za pomenutu grupu sa najvećom preciznošću. *A3C* model je pokazao solidne rezultate i evaluiran je empirijski, gde uočavamo da je robot u većini slučajeva uspešno paralelno pratio prvi red jagoda, zatim najčešće uspešno napravio "polukružno skretanje" i na kraju se uspešno kretao između prvog i drugog reda jagoda.

Osim datih rezultata, bitno je napomenuti da je u ovom radu dokazano da je moguće obučiti veštačko-inteligentni model bez velikih novčanih investicija i bez kreiranja bilo kakvog *hardware*-a, putem simuliranja u besplatnim *Unity* alatima. Dati rad predstavlja dobru osnovu ne samo za dalji razvoj datog "virtuelnog" 3D agenta, već i za razvoj stvarnog robota koji bi izvršavao takav ili slične zadatke u poljoprivredi.

5. LITERATURA

Svim navedenim linkovima je pristupljenu u junu 2020.

[1] https://unity.com/

[2] A. Juliani, V-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, D. Lange. Unity: A General Platform for Intelligent Agents, 2018

[3] https://github.com/tensorflow/models/tree/master/-research/object_detection

[4] https://colab.research.google.com/notebooks/-intro.ipynb

[5] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K., 2016, June. Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937).

[6] https://www.tensorflow.org/

[7] http://wiki.fast.ai/index.php/Fine_tuning

[8] http://cocodataset.org/

Kratka biografija:

Timotej Orčić rođen je u Novom Sadu 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Neuronske mreže i Metode poslovne inteligencije odbranio je 2020. god.
kontakt: timotej.orcic@gmail.com