SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Tomislav Smolčić

PROUČAVANJE ASTEROIDA U PREGLEDIMA NEBA PRIMJENOM NEURONSKIH MREŽA

Diplomski rad

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA I INFORMATIKA; SMJER NASTAVNIČKI

Tomislav Smolčić

Diplomski rad

Proučavanje asteroida u pregledima neba primjenom neuronskih mreža

Voditelj diplomskog rada: izv. prof. dr. sc. Goranka Bilalbegović

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2018.

Prvenstveno se zahvaljujem svojoj mentorici, na strpljenju i podršci. Najveća hvala mojoj najužoj obitelji na razumijevanju i mojim prijateljima i kolegama koji su mi uljepšali ove studentske godine.

Sažetak

Asterodi pružaju neusporediv pogled u prošlost Sunčevog sustava, ali i predstavljaju potencijalnu opasnost za planet Zemlju. Detekcija i identifikacija asteroida od iznimne su važnosti. Zbog toga su i danas u tijeku astronomska istraživanja čiji je cilj otkrivanje novih i praćenje orbita već poznatih, ali potencijalno opasnih asteroida. U takvim istraživanjima generiraju se ogromne količine podataka zbog čega je važan razvoj računalnih metoda za njihovu brzu i točnu obradu.

U ovom radu predlažemo neuronske mreže, algoritme strojnog učenja, za detekciju asteroida na fotografijama iz Sloanovog digitalnog pregleda neba (SDSS, engl. *Sloan Digital Sky Survey*) kao mogući smjer u razvoju računalnih sustava za obradu podataka dobivenih u astronomskim istraživanjima. Uspoređujemo učinak klasične i konvolucijske neuronske mreže koje su izvedene u programskom jeziku Python i njegovim paketima za strojno učenje.

U Dodatku, kao metodički dio diplomskog rada, predstavljena je uporaba programskog jezika Python za obradu slika kao nastavna jedinica iz predmeta Informatika u prirodoslovno-matematičkim gimnazijama.

Ključne riječi: asteroidi, strojno učenje, neuronske mreže, programski jezik Python, Scikit-Learn, TensorFlow

Study of asteroids in digital sky surveys using neural networks

Abstract

Asteroids give a specific insight into history of the Solar system, but also present a potential danger for the planet Earth. It is very important to detect and identify asteroids. Because of that, there are ongoing research projects with the purpose of finding new asteroids and tracking those already known, but potentially dangerous. In this research large amounts of data are generated. Therefore it is important to develop new computer methods to process data in the extremely fast and accurate way.

In this thesis we propose neural networks, machine learning algorithms, for detecting asteroids in data from the Sloan Digital Sky Survey (SDSS). This is a possible direction for development of computational methods to process data gathered in astronomy. We compare performance of standard and a convolutional neural networks implemented in Python programming language and its machine learning libraries.

In the Appendix, image processing in Python is presented as a topic for high school computer science classes.

Keywords: asteroids, machine learning, neural networks, Python programming language, SciKit-Learn, TensorFlow

Sadržaj

1 Uvod **1**

2 Asteroidi i Sloanov digitalni pregled neba **3**

2.1 Otkrivanje asteroida 5

2.2 Sloan Digital Sky Survey (SDSS) 5

2.2.1 SDSS-ova digitalna kamera 6

2.2.2 Asteroidi u SDSS-u 9

3 Osnove strojnog učenja **11**

3.1 Neuronske mreže 12

3.1.1 Perceptron 13

3.1.2 Višeslojni perceptron (MLP) 15

3.1.3 Treniranje neuronskih mreža 16

3.1.4 Konvolucijske neuronske mreže 17

4 Istraživanje asteroida neuronskim mrežama **19**

4.1 Skup podataka 19

4.1.1 Veličina 21

4.1.2 Podjela 22

4.2 Klasična neuronska mreža 23

4.2.1 Omjer nijansi boja 24

4.2.2 Klasteri boja 26

4.2.3 Implementacija 28

4.2.4 Rezultati 29

4.3 Konvolucijska neuronska mreža 30

4.3.1 Implementacija 30

4.3.2 Rezultati 31

5 Zaključak **33**

Dodatak **36**

A Obrada slika u Pythonu **36**

Literatura **42**

1 Uvod

Detekcija i identifikacija asteroida od iznimne je važnosti. Kao ostaci protoplanetarnog diska asteroidi pružaju neusporediv pogled u prošlost Sunčevog sustava. Svaka NASA-ina misija prema vanjskim planetima Sunčevog sustava, ukoliko je moguće, uključuje prolaz letjelice pokraj nekog malog tijela Sunčevog sustava. O njihovoj građi se još uvijek puno toga ne zna te se takve prilike za proučavanje asteroida, kometa i drugih malih tijela ne propuštaju.

Važnost detekcije malih tijela zbog mogućnosti sudara ili bliskog prolaska kraj Zemlje nije potrebno naglašavati. Veliki broj asteroida s kojima je Zemlja imala bliski susret u nedavnoj prošlosti detektiran je tek nakon što su oni već prošli kroz naše susjedstvo. Iz tog razloga istraživanja koja direktno ili indirektno proučavaju Sunčev sustav dobivaju sve veću pozornost.

U početku istraživanja asteroida, osim njihove detekcije, najveći problem i zadatak astronoma bio je točno izračunati orbite i moći ih pouzdano pratiti. Danas su matematičke tehnike proširene do granice kada to nije problem, a tehnologija omogućava predviđanje orbita asteroida gotovo neograničeno u budućnost te uvid u njihovu prošlost i porijeklo.

Astronomska istraživanja gotovo uvijek proizvode ogromne količine podataka. *Large Synoptic Survey Telescope* (LSST) [1] će krenuti s radom 2021. godine i svojim opsegom na neki način naslijediti misiju *Sloan Digital Sky Survey* (SDSS) [2]. To se naročito odnosi na područje fotometrijskih opažanja gdje se od LSST-a, zbog njegove 3.2-gigapikselne digitalne kamere, očekuju izvrsni rezultati. LSST će generirati 20 TB podataka svake noći, ukupno 60 PB (petabajta) tijekom desetogodišnje misije! Specifičnost ovog istraživanja je da će svi podaci koje teleskop skupi biti dostupni u skoro realnom vremenu. Cilj istraživanja je da obavijesti o detekcijama kratkotrajno vidljivih objekata budu odaslane znanstvenoj zajednici unutar 60 sekundi od njihove detekcije.

Takve količine podataka i ambiciozni ciljevi zahtjevaju razvoj novih, automatiziranih metoda analize podataka sposobnih s velikom točnošću detektirati i identificirati različite astronomske objekte. Metode strojnog učenja pokazuju se kao vrlo uspješan pristup u takvim problemima. Naročito su značajne moderne neuronske mreže, koje su zadnjih dvadesetak godina doživjele strelovit razvoj i danas su, kao

i cijelo područje strojnog učenja i umjetne inteligencije, jedno od najzanimljivijih područja računarskih znanosti, s primjenama u svim poljima znanosti i tehnologije.

U drugom poglavlju ovog rada opisane su osnovne karakteristike asteroida, s naglaskom na detekciju sa Zemlje. Opisuje se način rada s podacima koje prikuplja SDSS istraživanje. U trećem poglavlju dan je kratki pregled povijesnog razvoja strojnog učenja i općenitog rada neuronskih mreža te specijalno konvolucijskih neuronskih mreža. U četvrtom poglavlju opisuje se skup podataka korišten u radu i implementacija dviju neuronskih mreža za detekciju asteroida. U Dodatku A izložen je prijedlog nastavne jedinice u prirodoslovno-matematičkim gimnazijama: ”Obrada slika u Pythonu”.

2 Asteroidi i Sloanov digitalni pregled neba

Asteroidi su vjerojatno najpoznatija vrsta objekata iz skupine tzv. *malih tijela Sunčevog sustava* (engl. *Small Solar System bodies*). Međunarodna astronomska unija [3] (engl. IAU, *International Astronomical Union*) definira malo tijelo Sunčevog sustava kao svako tijelo koje kruži oko Sunca, a nije planet, patuljasti planet ili prirodni satelit. Tako u ovu skupinu spadaju još i kometi, trans-neptunski objekti itd.

Međutim, prvi asteroidi otkriveni su tek početkom 19. stoljeća. Za razliku od kometa koji su jasno vidljivi kako se približavaju Suncu i postaju aktivni, asteroidi su većinom nevidljivi golim okom. Nemaju vlastitog sjaja, većina je veoma malena i čak kada se vide na nebu u većini slučajeva izgledaju kao veoma blijede zvijezde.

Krajem 18. stoljeća, njemački znanstvenik Johan Daniel Titius formulirao je hipotezu koja je kasnije postala poznata kao Titius-Bodeov zakon. Johan Elert Bode je znanstvenik koji je prihvatio Titusovu ideju po kojoj je velika poluos orbite svakog planeta zadana matematičkim izrazom:

$$a = 4 + n, \qquad n = 0, 3, 6, 12, 24, 48, 96, 192, 384. \tag{2.1}$$

Kada se ovaj izraz primijeni na Sunčev sustav, dobiju se prilično precizna predviđanja velikih poluosi poznatih planeta (u astronomskim jedinicama, uvećana za faktor 10). Danas je poznato da zakon nema teorijsku podlogu i da se navjerojatnije radi o slučajnosti. U doba formuliranja Titius-Bodeov zakon je bio prihvaćen u znanstvenoj zajednici, pogotovo nakon što je 1781. godine otkriven Uran na udaljenosti koju predviđa ovaj zakon.

Titius-Bodeov zakon također predviđa postojanje planeta između Marsa i Jupitera. U svrhu njegovog pronalaska pokrenut je vjerojatno prvi međunarodni istraživački projekt nazvan "*Himmelspolizey*" (njemački Himmel - nebo i Polizei - policija). I zaista, 1801. godine, talijanski astronom Giuseppe Piazzi, inače nepovezan s istraživanjem nebeske policije, pronašao je planet na predviđenoj udaljenosti. Isprva je mislio da se radi o novoj zvijezdi. Međutim, uspio je pratiti objekt kako se pomiče nekoliko uzastopnih noći. Kako je bio upoznat s potragom za novim planetom koja je tada trajala, odlučio je poslati svoja mjerenja vodećim znanstvenicima tog istraživanja.

Njemački matematičar Carl Friedrich Gauss uspio je, iz svega nekoliko Piazzije-

vih mjerenja, odrediti orbitu objekta, i to točno na predviđenoj udaljenosti, između Marsa i Jupitera. Skoro punu godinu nakon Piazzijevog otkrića, Heinrich Wilhelm Olbers uspio je ”ponovno otkriti” nestali planet, kojemu je Piazzi dao ime Ceres. Ubrzo su na sličnoj udaljenosti pronađena još tri planeta, Pallas, Juno i Vesta, a nakon 40-ak godina i peti planet nazvan Astraea. S vremenom, počelo se sumnjati da je moguće da se toliko planeta nalazi na tako bliskim orbitama i postalo je jasno da je potrebna nova klasifikacija. Stvorena je nova grupa objekata, nazvana *asteroidi*. Međunarodna astronomska unija je 2006. godine znatno izmijenila sustav klasifikacije tijela u Sunčevom sustavu i Ceres je ”promoviran” u potpuno novu grupu objekata, patuljasti planet.

Većina asteroida nalazi se u tzv. *glavnom pojasu*, između Marsa i Jupitera. Upravo zbog blizine Jupitera i njegovog gravitacijskog utjecaja u tom području nije stvoren planet. Jupiterova gravitacija uzrokovala je velik broj sudara među objektima koji su se formirali unutar pojasa. To je spriječilo njihovo okrupnjavanje kao što je slučaj kod postanka planeta Sunčevog sustava.

Osim u glavnom pojasu, asteroidi se nalaze i u drugim područjima Sunčevog sustava. Najbliži Suncu su tzv. *Near Earth Asteroids* (skraćeno NEA), grupa asteroida čija orbita prolazi blizu Zemljine, po čemu su i dobili ime. Prvi asteroid otkriven iz te grupe je Eros. Asteroidi unutar NEA grupe mogu se jako razlikovati; neki kruže u potpunosti unutar Zemljine orbite, dok orbite nekih sežu i dalje od glavnog pojasa asteroida, ali u jednom dijelu orbite približavaju se Zemlji.

Izgaranje asteroida veličine pikule u Zemljinoj atmosferi vidi se kao svijetla pruga na noćnom nebu. Nešto veći asteroid može se raspasti u atmosferi i proizvesti veoma jake eksplozije. Najnoviji primjer je asteroid koji se raspao iznad grada Chelyabinska, blizu Urala u Rusiji, 2013. godine. Eksplodirao je na visini od 35 km snagom deset puta većom od atomskih bombi iz Drugog svjetskog rata. Iznad rijeke Tunguska u Sibiru, 1903. godine, asteroid ili komet je eksplodirao i uništio ili srušio stabla u radijusu od 30 km. Asteroid slične veličine prošao je 2013. godine na udaljenosti od 27 700 km iznad površine Zemlje.

Udari asteroida veličine mjerene u kilometrima vrlo vjerojatno bi izazvali globalnu štetu, bilo podizanjem ogromnih oblaka prašine koji bi utjecali na klimu, ili tsunamijima koji bi uništili obale kontinenata uz ocean u kojem se udar dogodio. U svakom slučaju, prilično je očita važnost otkrivanja asteroida i pomno praćenje

njihovih orbita.

2.1 Otkrivanje asteroida

Detekcija asteroida je teška zbog činjenice da sa Zemlje izgledaju kao zvijezde, još k tome veoma slabog sjaja. Svako promatranje neba moralo se uspoređivati s kartama neba i katalozima zvijezda. Očito, ključna razlika je da su asteroidi puno bliže Zemlji od zvijezda. Za razliku od prividno mirujućih zvijezda, asteroidi se kreću po nebeskoj sferi.

Sredinom 19. stoljeća, u astronomiji se počela koristiti fotografija, tada nova tehnologija. Usporedbom dviju fotografija nastalih u razmaku od sat ili dva tražile su se ”zvijezde” koje su promijenile položaj između fotografija. Više nije bilo potrebno pretraživati kataloge ili stare karte neba.

S vremenom, pojavili su se strojevi koji su olakšavali proces traženja asteroida i kometa, temeljeni na ideji uspoređivanja dvaju fotografija. Danas se u tu svrhu koriste moderne kamere s CCD (engl. *Charge-Coupled Device*) senzorima. Poznato je preko 750000 asteroida i svaki dan se otkrivaju novi.

Danas se smatra da su asteroidi ključni za otkrivanje rane povijesti Sunčevog sustava. Oni su ono što je ostalo nakon formiranja planeta unutarnjeg Sunčevog sustava, a to je velika motivacija za pronalazak novih asteroida. Naravno, vrlo je značajna potraga za asteroidima koji bi mogli predstavljati potencijalnu opasnost za Zemlju. Tako postoji mnogo istraživanja čiji je cilj pronaći čim više asteroida iz NEA grupe. Američki kongres je 1998. godine naredio NASA-i da mora pronaći preko 90% asteroida iz NEA grupe. U tu svrhu pokrenuto je više istraživanja: Lowell Observatory Near-Earth-Object Search (LONEOS) koji je otkrio preko 19 tisuća asteroida (678 NEA), Catalina Sky Survey, Spacewatch, Near-Earth Asteroid Tracking (NEAT), Lincoln Near-Earth Asteroid program (LINEAR) [4]. Naravno, asteroidi su otkrivani i u istraživanjima kojima to nije primarna misija, kao što su Pan-STARRS (Panoramic Survey Telescope and Rapid Rensponse System) ili SDSS (Sloan Digital Sky Survey).

2.2 Sloan Digital Sky Survey (SDSS)

Sloan Digital Sky Survey je napravio najdetaljniju trodimenzionalnu mapu svemira do sad, s dubokim višebojnim slikama trećine neba i spektrima preko tri milijuna

astronomskih objekata [2].

Prve slike neba SDSS je proizveo 1998. godine, nakon deset godina izgradnje. Od tada do danas, opseg i cilj istraživanja su se mijenjali kroz četiri generacije SDSS-a. Dok je originalno SDSS koristio samo 2.5-metarski teleskop u Apache Point opservatoriju u Novom Meksiku, danas uz njega koristi i Irénée du Pont teleskop u Las Campanas opservatoriju u Čileu.

Sadašnja generacija istraživanja, SDSS-IV, podatke prikuplja isključivo u spektroskopskom modu. Do 2020. godine, u tijeku će biti tri velika istraživanja: APOGEE-2 je spektroskopsko istraživanje zvijezda Mliječne staze, eBOSS kozmološko istraživanje kvazara i galaksija te MaNGA, spektralna mjerenja oko 10 tisuća "obližnjih" galaksija.

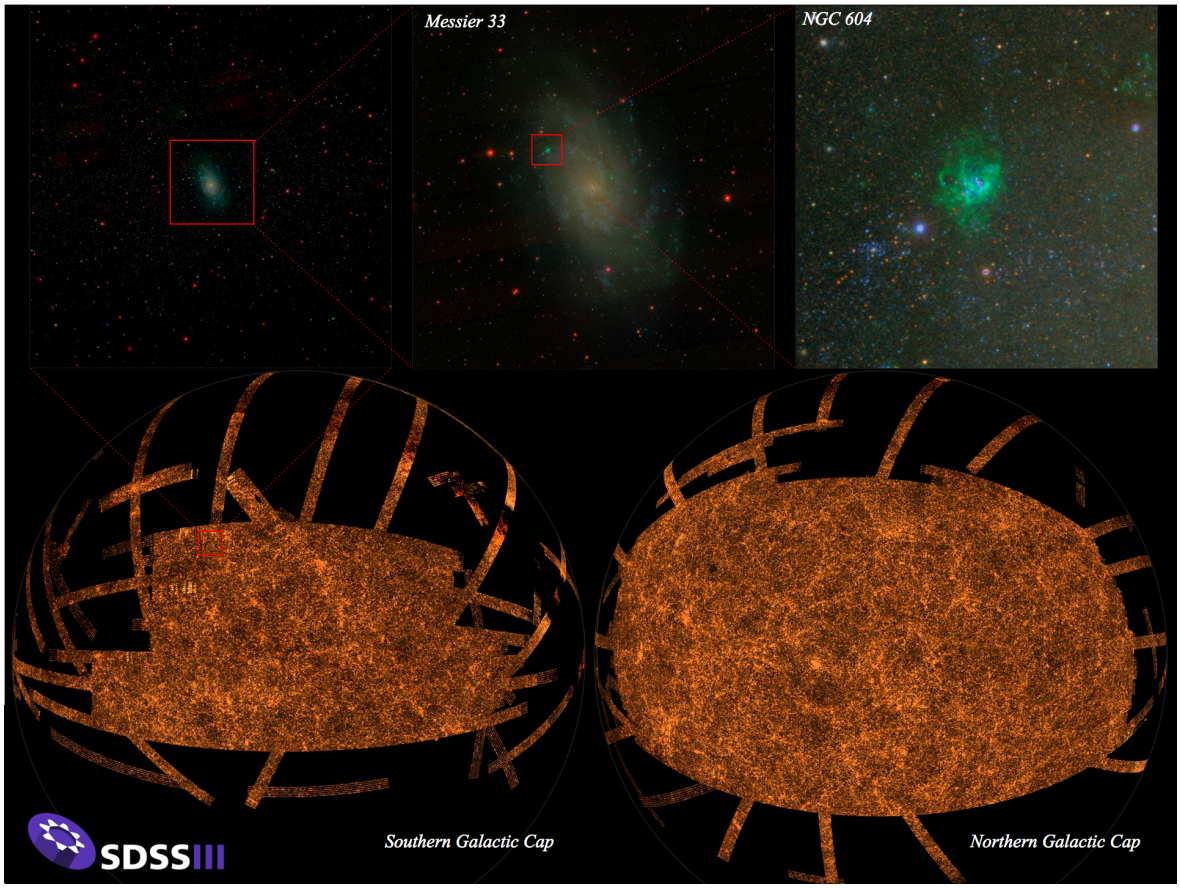
2.2.1 SDSS-ova digitalna kamera

Do 2011. godine SDSS-ov teleskop u Apache Pointu radio je s digitalnom kamerom, koja je 1998. godine, kada je teleskop počeo s radom, bila najveća digitalna kamera na svijetu. Od 1998. do 2008. godine kamera je snimila trećinu cijelog neba, i od milijuna fotografija, svake veličine 2.8 megapiksela, napravljena je najveća fotografija u boji neba do sad (slika 2.1) [5]. Kamera je od 2011. godine dio stalne kolekcije Smithsonian instituta kao priznanje njenog doprinosa astronomiji, ali sve fotografije dostupne su u SDSS-ovoj bazi podataka.

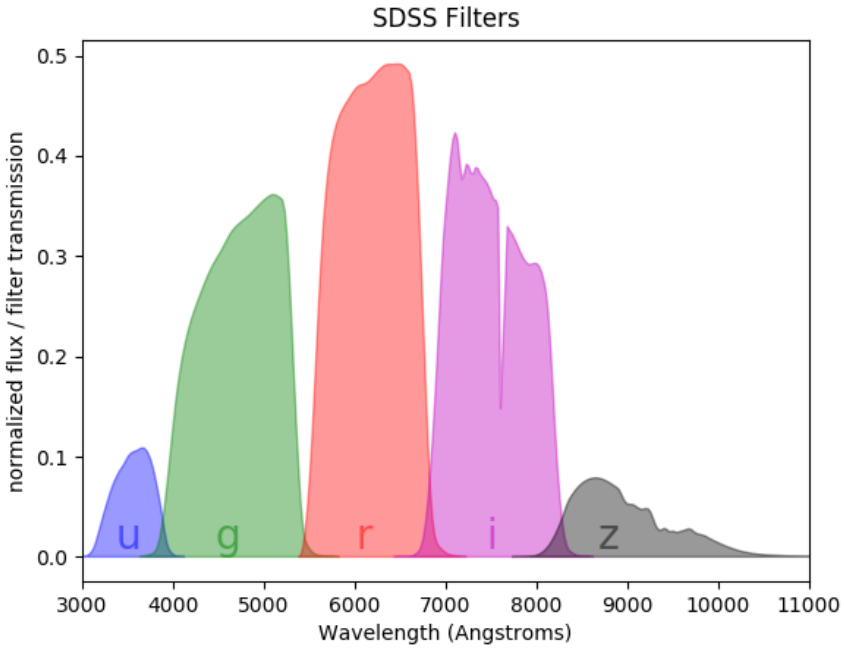
Kamera je prikupljala fotometrijske podatke koristeći mrežu trideset 2048×2048 pikselnih CCD senzora, poslaganih u šest stupaca po pet CCD-ova. Koristi se pet filtera za različita područja elektromagnetskog spektra, čije su centralne valne duljine prikazane u tablici 2.1, a propusnost filtera na slici 2.2. Na taj način kamera proizvodi pet slika svakog objekta, sve iz istog stupca CCD-ova, po jedna slika za svaki CCD, odnosno filter. Na slici 2.3 prikazan je raspored filtera i CCD senzora kamere.

u	3551 Å
g	4686 Å
r	6166 Å
i	7480 Å
z	8932 Å

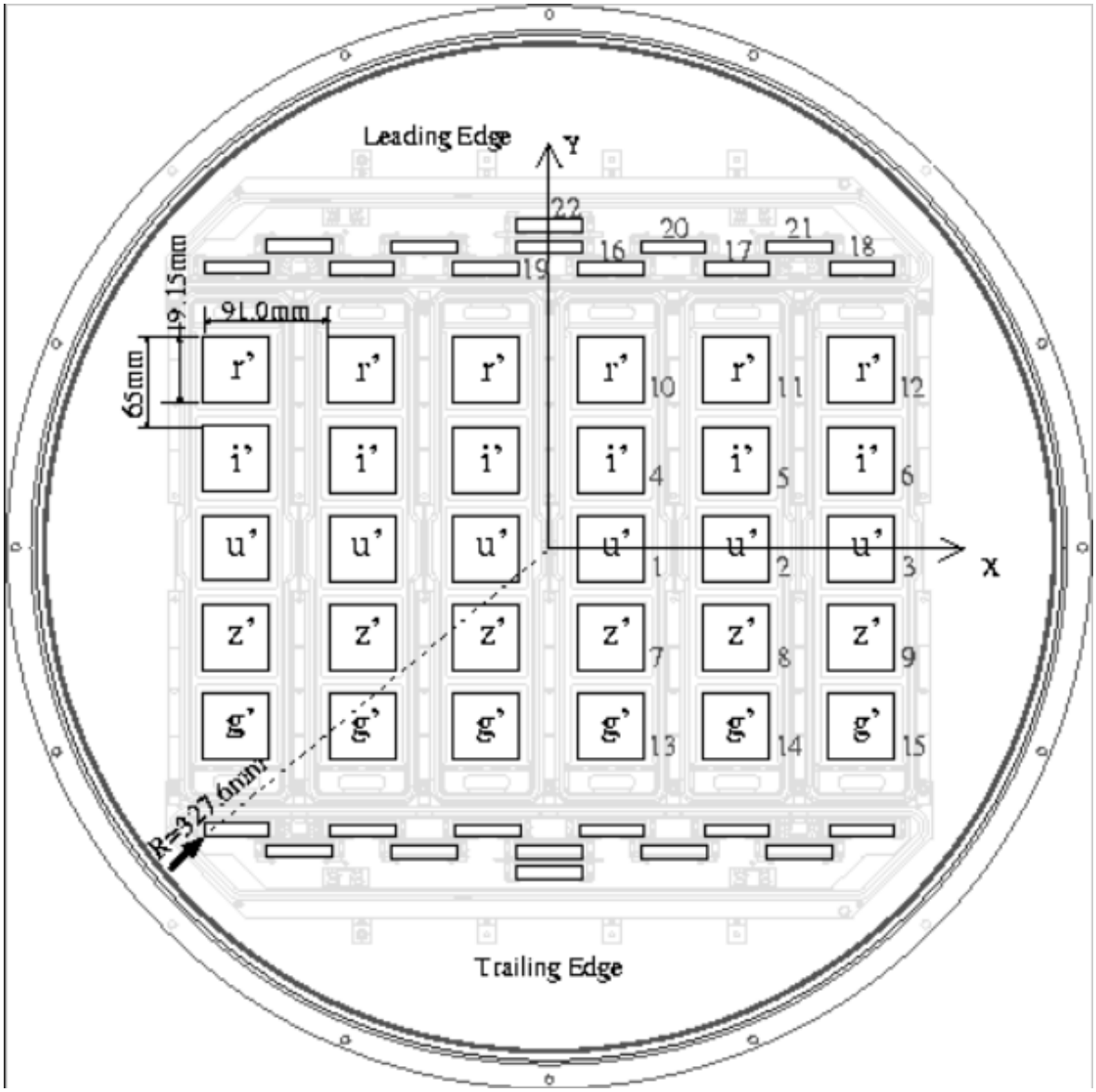
Tablica 2.1: Filteri i pripadne centralne valne duljine SDSS-ove kamere



Slika 2.1: Ilustracija različitih podataka dostupnih u SDSS-III. Gore lijevo je mala slika neba s galaksijom Messier 33 u sredini, slika gore u sredini i desno od nje su uvećani prikazi iste galaksije. Dvije slike u drugom redu prikazuju potpunu mapu neba dobivenu iz SDSS-III slike. (M. Blanton i SDSS kolaboracija, 2011.)



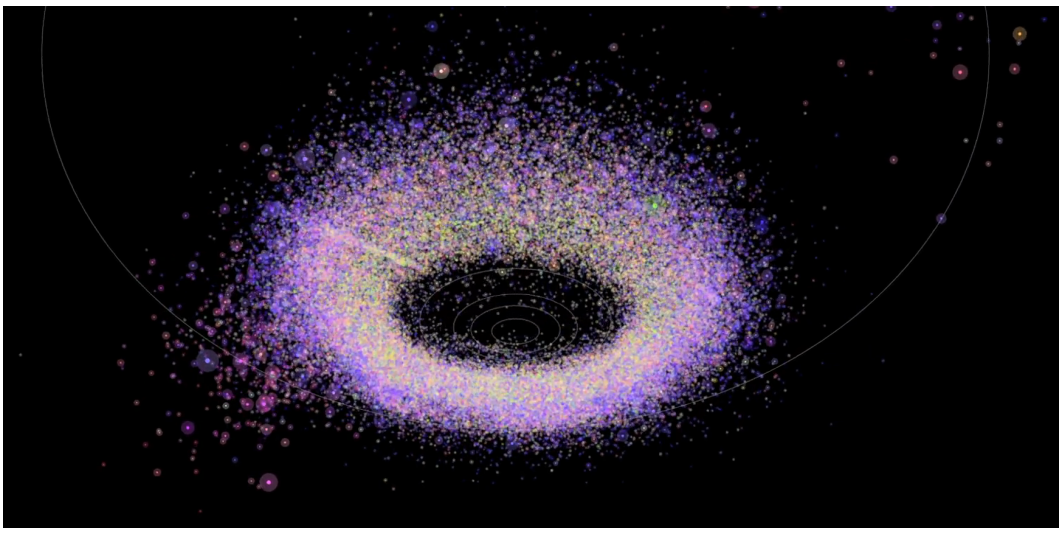
Slika 2.2: Propusnost filtera SDSS-ove kamere [6]



Slika 2.3: Prednji pogled na raspored filtera SDSS-ove kamere [7]

2.2.2 Asteroidi u SDSS-u

SDSS je identificirao preko 100 tisuća asteroida i ostalih malih tijela Sunčevog sustava. Slika 2.4 (Alex H. Parker, 2014.) prikazuje vizualizaciju 100 tisuća asteroida u SDSS-ovom katalogu, SDSS Moving Object Catalog (SDSSMOC) [8].

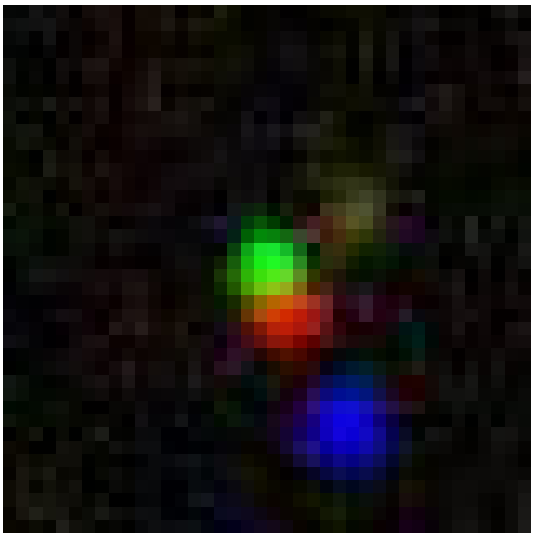


Slika 2.4: Vizualizacija asteroida u SDSS-ovom katalogu

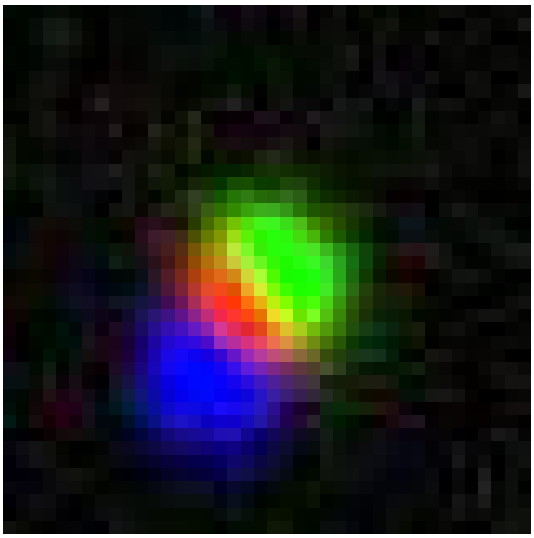
SDSS-ova istraživanja asteroida, od kojih je većina iz glavnog pojasa, dovela su do novih spoznaja o distribuciji veličine asteroida u glavnom pojasu i vezi između obitelji asteroida i njihovoj boji. Asteroidi u SDSS-ovim fotografijama mogu se identificirati zato što mijenjaju položaj tijekom vremena ekspozicije SDSS-ove kamere. Kako filteri (tablica 2.1 i slika 2.2) slikaju isto područje neba sekvencijalno, objekt koji se giba bit će na različitim položajima na fotografiji dobivenoj svakim filterom.

Primjer slike asteroida vidi se na slici 2.5. Jasno se vide odvojene tri grupe (u kasnijem tekstu, *klastera* ili *bloba*) različitih boja. Kako slike svih filtera nastaju u različitim trenucima, asteroid se u svakom trenutku nalazi na drugom mjestu, u odnosu na prividno mirujuće udaljene objekte, zvijezde i galaksije. Iako kamera koristi pet filtera, fotografije kao što je slika 2.5 nastaju sklapanjem podataka iz tri filtera, koji pokrivaju vidljivi dio elektromagnetskog spektra: *r, i, g*. Filter *i* pokriva najveće valne duljine (tablica 2.1) pa daje crvenu boju, *r* filter zelenu i *g* filter plavu boju.

Na slici 2.5 također se vidi da su crveni i zeleni klaster blizu dok je plavi klaster udaljeniji. Razlog leži u rasporedu filtera SDSS-ove kamere, što se dobro vidi na slici 2.3: *r* i *i* filter su jedan ispod drugoga, dok je *g* filter na dnu stupca. Dakle, više vremena prođe između nastanka slika filtera *i* i *g* nego između slika filtera *r* i



Slika 2.5: Asteroid na fotografiji SDSS-a



Slika 2.6: ”Sporiji” asteroid

i. Zbog toga, asteroid stigne prevaliti veći put između filtera *i* i *g* te je plavi klaster udaljeniji. Ukoliko je asteroid nešto sporiji (bilo stvarno sporiji ili prividno zbog veće udaljenosti), doći će do većeg preklapanja zelenog i crvenog klastera (npr. slika 2.6). Na ovom primjeru čak dolazi i do preklapanja crvenog i zelenog klastera. Na područjima preklapanja, naravno dolazi do miješanja boja i pojave žute, odnosno ljubičaste boje.

3 Osnove strojnog učenja

Strojnim učenjem danas se rješavaju problemi za koje se donedavno mislilo da su nemogući ili prekompleksni za tradicionalno programiranje. Metode strojnog učenja koriste se npr. za ranu procjenu rizika od razvoja kardiovaskularnih bolesti [9], predviđanje optimalnih putanja zrakoplova [10], za otkrivanje egzoplaneta [11], u istraživanjima fizike plazme [12] te u mnogim drugim područjima znanosti i tehnologije.

Strojno učenje je primjena numeričkih algoritama koji poboljšavanju svoj *učinak* na danom *zadatku* obzirom na *iskustvo* [13]. *Zadatak* je predvidjeti numeričku vrijednost na temelju danih numeričkih unosa.

U problemima koji su tradicionalnim programiranjem nerješivi, izuzetno kompleksni, ili jednostavno zahtijevaju dugačka i kompliciranа algoritamska rješenja, metode strojnog učenja se pokazuju iznimno učinkovite. Ti algoritmi se uglavnom temelje na jako velikom broju parametara koji za ulazne podatke daju dobru procjenu rješenja. Prednost je da te parametre ne treba ni na bilo koji način unaprijed odrediti, što omogućuje rješavanje problema čak i kada nisu poznati svi njegovi detalji niti približan oblik rješenja. Za tako nešto u većini slučajeva je potrebna velika količina podataka, što je jedan od razloga brzog razvoja strojnog učenja u posljednjih desetak godina.

Počeci strojnog učenja datiraju iz pedesetih godina 20. stoljeća s izumom *perceptrona* [14], algoritma za binarnu klasifikaciju, koji je prvotno bio namijenjen prepoznavanju uzoraka na slikama. Od perceptrona se mnogo očekivalo, poznati američki dnevnik *New York Times*, 1958. godine, navodi da je perceptron ”embrio elektroničkog računala za koje američka mornarica vjeruje da će moći hodati, govoriti, vidjeti, pisati, razmnožavati se i biti svjesno svojeg postojanja.” [15]. Ubrzo se pokazalo da sustavi temeljeni na perceptronu ne mogu biti trenirani da prepoznaju neke klase uzoraka [16], što je dovelo do višegodišnjeg zastoja u razvoju strojnog učenja. Povezivanjem statistike i računalnih znanosti u osamdesetim godinama 20. stoljeća, dolazi do razvoja probabilističkog pristupa strojnom učenju. S iznimno velikim količinama podataka na raspolaganju, pojavila se mogućnost razvoja sustava koji mogu analizirati i *učiti* iz tih podataka.

Strojno učenje, po načinu treniranja, danas se dijeli u nekoliko kategorija: *nad-*

zirano, *nenadzirano* učenje te podupirano učenje (engl. *reinforcement learning*). U nadziranom učenju postoji skup podataka za koji je poznato rješenje i zadatak algoritma je naučiti na temelju toga skupa i naći rješenje za nove, neviđene podatke. Dobar primjer je detekcija objekata na slikama: algoritmu možemo dati neki broj slika na kojima je ručno označen objekt koji želimo detektirati, algoritam treniramo na tom skupu slika i zatim očekujemo da će moći detektirati isti objekt na novoj slici.

Kod nenadziranog učenja ne postoji takav trening set. Cilj algoritama nenadziranog učenja je samostalno otkriti strukturu ili distribuciju podataka, bez čovjekovog nadzora i prethodno poznatih rješenja. Upravo zbog činjenice da podaci ne dolaze s poznatim oznakama, nemoguće je trenirati algoritam kao što bi to radili u slučaju nadziranog učenja. Također, teško je ocijeniti koliko dobro algoritam radi. Međutim, postoje situacije kada su algoritmi nenadziranog učenja korisni. Često je potrebno neki skup podijeliti u više grupa podataka međusobno sličnih po nekom kriteriju. Primjer je grupiranje kupaca u nekoj internetskoj trgovini. Algoritam može primjetiti da su korisnici koji su kupili određeni proizvod slične dobi, ili ih zanimaju slični proizvodi. Općenito, algoritmi nenadziranog učenja mogu se podijeliti u algoritme klasteriranja (grupiranje, engl. *clustering*) gdje se podaci razvrstavaju u klastere te algoritme gdje se podaci raspoređuju po kontinuiranim vrijednostima.

Podupirano učenje se odvija na principu nagrade i kazne. Algoritmu podupiranog učenja je poznat cilj te mogući potezi prema cilju. Svaki potez se nekako vrednuje, bilo pozitivno ili negativno. Dobar primjer su igre, kao šah ili Go, u kojima su sustavi strojnog učenja u zadnjih nekoliko godina pokazali veliki napredak [17]. U igrama su pravila i cilj dobro poznati i svaki potez koji vodi pobjedi je nagrađen, dok se očito, svaki potez koji vodi porazu kažnjava.

3.1 Neuronske mreže

Danas vjerojatno najpoznatiji i najnapredniji *sustavi* strojnog učenja su neuronske mreže. Ideja o neuronskim mrežama pojavila se 1943. godine, kada su neuropsiholog Warren McChulloh i matematičar Walter Pitts u svom članku [18] predstavili pojednostavljen računalni model načina rada biološkog neurološkog sustava.

Ugrubo, biološki neuroni se sastoje od tijela i mnoštva dugačkih produžetaka, dendrita. Jedan dendrit je puno duži i naziva se akson. Pomoću aksona neuroni se

medjusobno povezuju. Neuroni su veoma specijalizirane stanice za procesuiranje i prijenos signala. Kada jedan neuron primi signal od drugog neurona, ili više njih, on sam odašilje vlastiti signal neuronima s kojima je povezan. Naizgled, neuroni funkcioniraju na vrlo jednostavan način, ali neurološki sustav, koji se u čovjeka sastoji od nekoliko desetaka milijardi neurona, sposoban je za iznimno kompleksne operacije. Iako se o neurološkom sustavu još uvijek puno toga ne zna, čini se da su neuroni organizirani u slojeve [19], kako je prikazano na slici 3.1. Većina ovih svojstava na neki je način prenesena u strukturu i način rada neuronskih mreža.

3.1.1 Perceptron

McCulloh i Pitts su predložili jednostavan model, kasnije poznat kao *umjetni neuron*, koji ima više binarnih ulaznih podataka i jedan, također binarni izlaz. Pokazali su da već tako jednostavan model može graditi sustav sposoban za izvršavanje svih logičkih operacija.

Perceptron je zapravo najjednostavnija arhitektura *umjetne neuronske mreže*. Iako se bazira na nešto drugačijoj vrsti umjetnog neurona, nazvanog LTU (engl. *Linear threshold unit*), ideja ostaje ista. Za razliku od običnog neurona, u slučaju LTU-a, ulazni podaci nisu binarne vrijednosti, već mogu biti realni brojevi. Shema rada LTU-a prikazana je na slici 3.2. Svakom ulaznom podatku pridjeljena je njegova težina i LTU jednostavno računa težinsku sumu svojih ulaza:

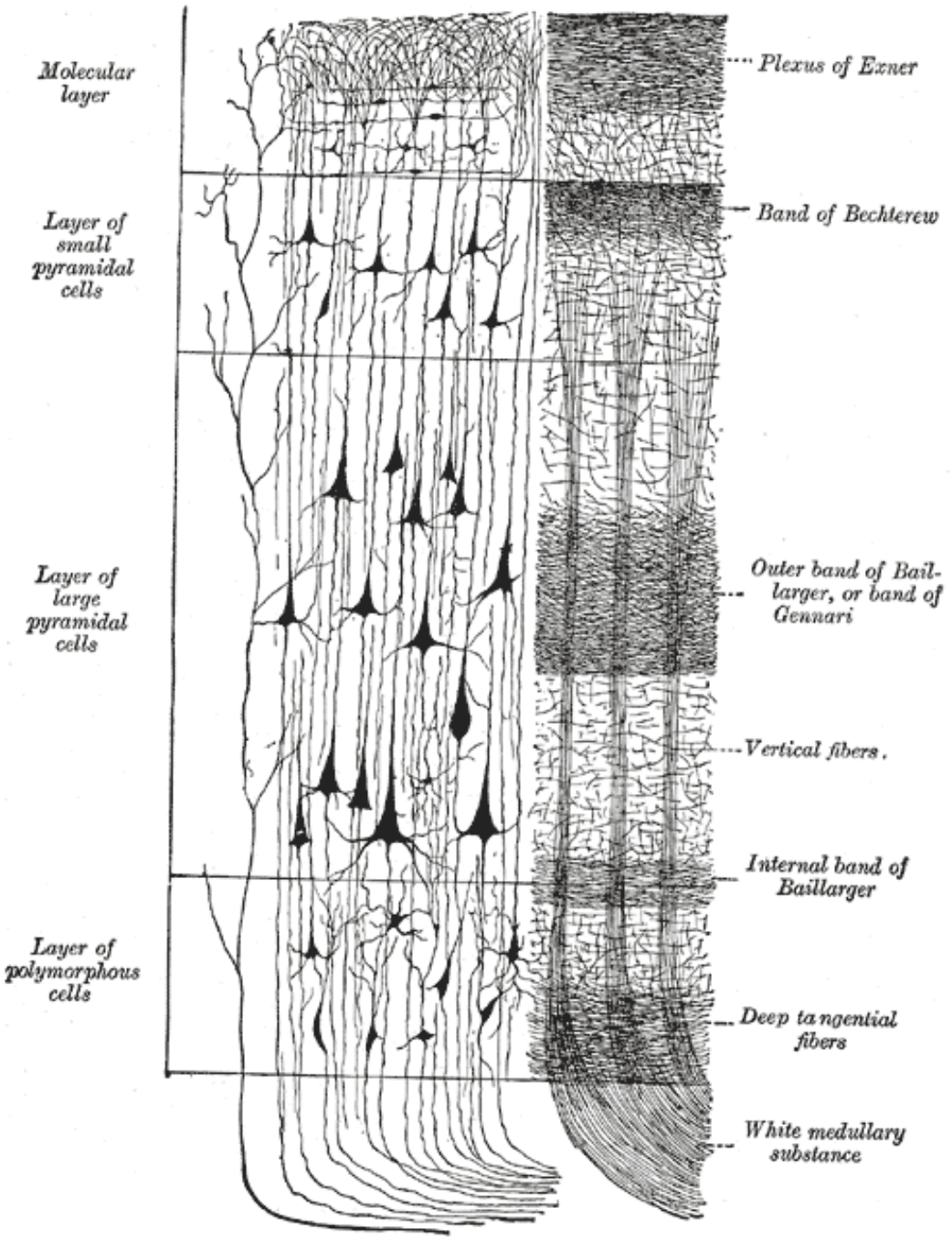
$$z = \sum w_i * x_i \tag{3.1}$$

gdje su x_i ulazni podaci, a w_i njihove pripadajuće težine. Ako se ulazni podaci x_i promatraju u obliku vektora $X = (x_1, x_2, ..., x_n)$ s pripadajućim težinama $W = (w_1, w_2, ..., w_n)$, jednadžba 3.1 se može zapisati i vektorski:

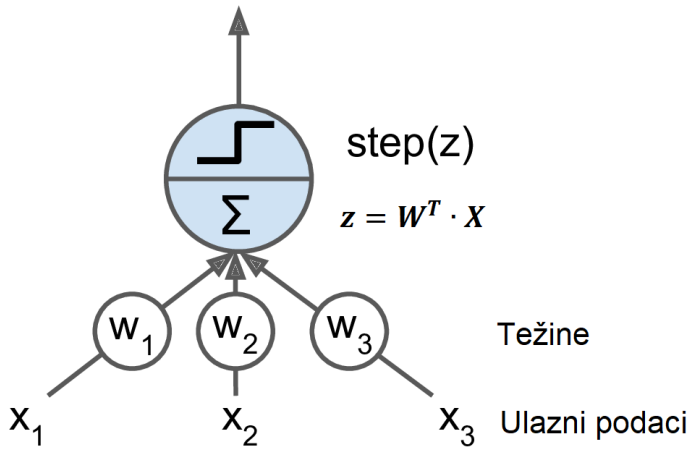
$$z = W^T \cdot X \tag{3.2}$$

Naposlijetku, na z se primjenjuje *step funkcija*, najčešće *Heavisideova* funkcija:

$$step(z) = \begin{cases} 0; z < 0 \\ 1; z \geq 0 \end{cases} \tag{3.3}$$



Slika 3.1: Čeoni režanj. Vidljiv je slojeviti raspored neurona.



Slika 3.2: Linear threshold unit [20]

LTU dakle računa linearnu kombinaciju ulaznih vrijednosti i daje pozitivan izlaz ako rezultat prelazi prag. Perceptron se sastoji od jednog sloja LTU-ova, od kojih svaki prima sve ulazne podatke, uz dodatak tzv. *bias* člana, posebnog neurona koji samo proslijeđuje konstantu, obično 1. *Bias* član se može shvatiti kao analogon slobodnom članu b u linearnoj funkciji $y = ax + b$.

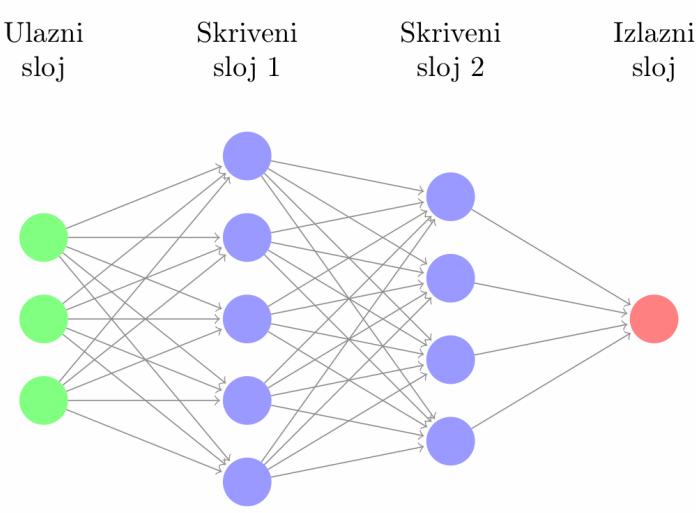
Perceptron se trenira tako da mu se proslijeđuju pojedinačni primjerci iz skupa za treniranje i za svaki primjer daje predviđanje u vidu izlazne vrijednosti. Predviđanje se uspoređuje s očekivanjem. Sukladno tome se prilagođavaju težinske vrijednosti, po jednadžbi 3.4, gdje je $w_{i,j}$ težina između i-tog neruona i j-tog izlaza, x_j je j-ti trening primjerak, $\hat{y}_j$ njegova očekivana vrijednost, y_j predviđena vrijednost, a η tzv. *stopa učenja* (engl. *learning rate*). Manja stopa učenja povećava vjerojatnost nalaska boljeg rješenja, ali usporava proces treniranja.

$$w_{i,j} = w_{i,j} + \eta(\hat{y}_j - y_j)x_j \tag{3.4}$$

3.1.2 Višeslojni perceptron (MLP)

Problemi perceptrona, na koje su u svom članku [16] ukazali Minsky i Papert, uzrokovali su višegodišnji zastoj u razvoju neuronskih mreža i fokus istraživanja se prebacio na metode strojnog učenja koje su davale bolje rezultate i imale bolju teorijsku podlogu. Međutim, pokazalo se da se većina problema može riješiti slaganjem neurona u više slojeva. Takvu arhitektura naziva se *višeslojni perceptron* (engl. MLP, *Multilayer Perceptron*)

MLP se sastoji od jednog sloja neurona koji proslijeđuju ulazne podatke (zbog čega se taj sloj naziva *ulazni sloj*) jednom ili više uzastopnih slojeva LTU-ova, od kojih se posljednji sloj naziva *izlazni sloj*. Najčešće je svaki neuron povezan sa svim neuronima iz prethodnog sloja. Višeslojni perceptron u biti je ono što danas nazivamo *umjetna neuronska mreža*, ili samo *neuronska mreža*. Neuronske mreže s više od jednog skrivenog sloja nazivaju se *duboke neuronske mreže* (slika 3.3).



Slika 3.3: Duboka neuronska mreža s dva skrivena sloja

3.1.3 Treniranje neuronskih mreža

Treniranje neuronskih mreža dugo je predstavljalo izazov, sve do 1986. godine kada je predstavljen *backpropagation* algoritam za treniranje neuronskih mreža [21]. Algoritam se sastoji od dvije faze. U prvoj fazi neuronska mreža za svaki primjerak iz skupa za treniranje računa predviđanje (prolazak unaprijed, engl. *forward pass*). Nakon toga računa pogrešku u odnosu na očekivani rezultat za taj primjerak iz skupa za treniranje. U drugoj fazi prolazi unatrag kroz mrežu i računa koliko svaki neuron doprinosi ukupnoj grešci i na temelju toga prilagođava težine (prolazak unatrag, engl. *reverse pass*).

U drugoj fazi algoritam računa grešku od izlaznog sloja prema ulaznom i propagira grešku, odakle i naziv algoritma. Naposlijetku, koristeći algoritam postupnoga opadanja (engl. *gradient descent*) algoritam korigira sve težinske parametre u mreži, koristeći prethodno izračunate greške. Kako step funkcija 3.3 nije diferencijabilna i zbog toga nije moguća primjena algoritma postupnoga opadanja, zamijenjena je

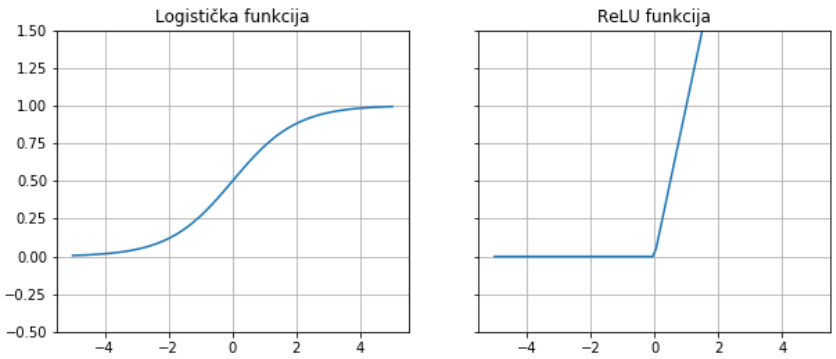
logističkom ili sigmoid funkcijom, 3.5.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.5}$$

Ubrzo su se počele koristiti i druge funkcije, kao što je tangens hiperbolni. U modernim neuronskim mrežama se koristi tzv. *ReLU* funkcija (3.6) i njene razne varijante.

$$ReLU(z) = max(0, z) \tag{3.6}$$

Iako ovakve funkcije nisu glatke, pokazuje se da u praksi to nije problem, a velika je prednost da su veoma jednostavne za računanje i zbog toga učinkom nadmašuju tradicionalno korištene funkcije [22]. Grafovi logističke i ReLU funkcije prikazani su na slici 3.4.



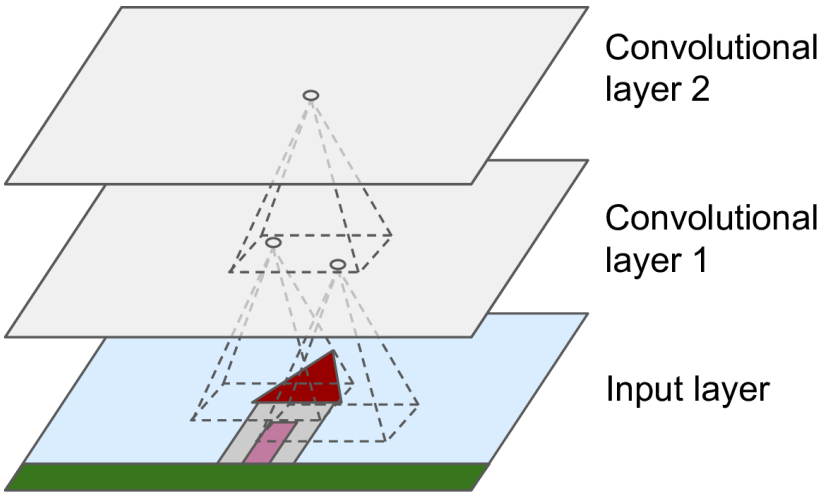
Slika 3.4: Grafovi logističke i ReLU funkcije

3.1.4 Konvolucijske neuronske mreže

S vremenom je došlo do razvoja specijaliziranih neuronskih mreža. Konvolucijske neuronske mreže (engl. CNN, *Convolutional neural networks*) [23] koriste se za probleme koji uključuju obradu kompleksnih vizualnih podataka, kao što je detekcija i klasifikacija objekata na slikama te autonomna vožnja. Ove mreže se veoma uspješno koriste i na području obrade prirodnog jezika (engl. NLP, *natural language processing*).

U usporedbi s klasičnim neuronskim mrežama, gdje su svi neuroni u susjednim slojevima u potpunosti međusobno povezani, kod konvolucijskih već jedan neuron može biti povezan samo s malim brojem neurona iz prethodnog sloja (slika 3.5). Ovo omogućuje da se neke male posebnosti iz prvog skrivenog sloja izoliraju i slažu

u složenije uzorke u sljedećem sloju. Takvi slojevi nazivaju se konvolucijski slojevi.



Slika 3.5: Konvolucijska mreža s dva konvolucijska sloja [20]

Druga nova vrsta skrivenih slojeva su slojevi sažimanja (engl. *pooling* slojevi), čiji je zadatak na neki način smanjiti ulaznu sliku. Slično kao što je slučaj kod konvolucijskih slojeva, neuroni slojeva sažimanja su povezani samo s manjim brojem neurona iz prethodnog sloja. Neuroni u sloju sažimanja obično računaju maksimum ili srednju vrijednost neurona s kojima su povezani. Na taj način se prosljeđuje bitno umanjena slika i smanjuju memorijski zahtjevi, koji su inače veliki problem konvolucijskih mreža.

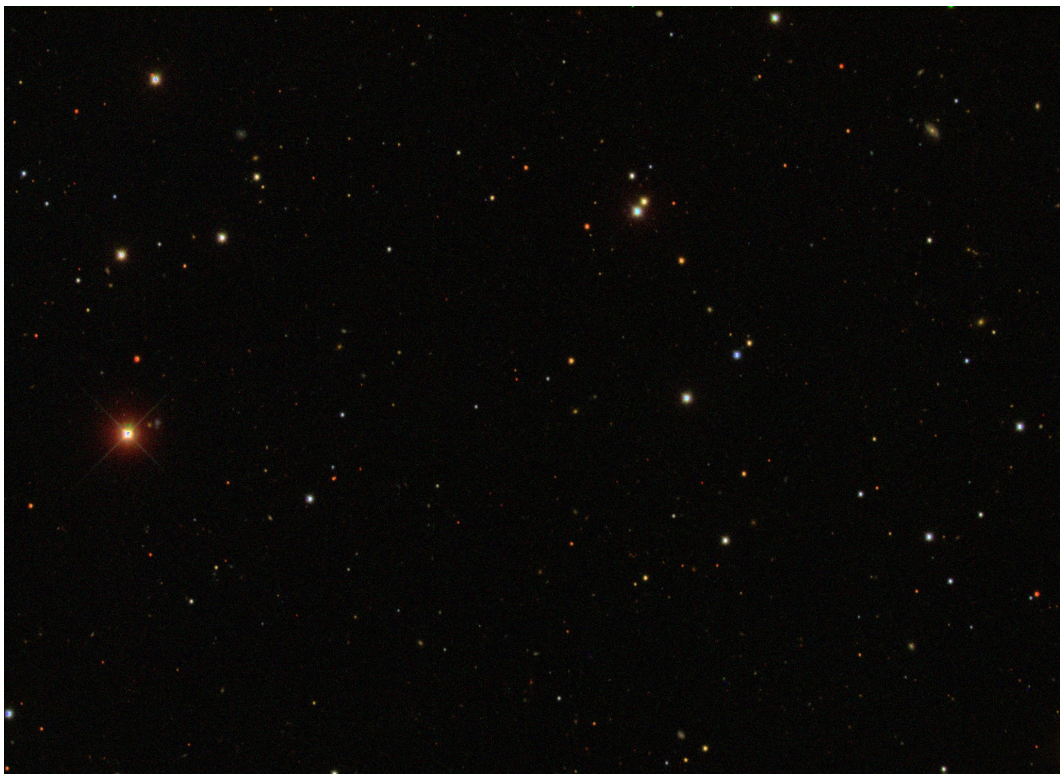
Konvolucijske mreže sastoje se od više konvolucijskih, slojeva sažimanja i običnih, potpuno povezanih slojeva. Veličina slojeva, njihov broj i redoslijed uvelike ovise o pojedinom slučaju: dimenzijama ulazne slike i sl.

4 Istraživanje asteroida neuronskim mrežama

4.1 Skup podataka

Slike korištene za ovaj rad dolaze iz 12. izdanja SDSS-III kolaboracije (*SDSS-III DR12*), koje sadrži podatke prikupljene iz nekoliko istraživanja, zaključno s mjesecom srpnjem 2014. godine. SDSS-ovi podaci su dostupni besplatno, u vidu spektara, različitih mjerenja i potpunih slika noćnog neba u više digitalnih formata. Moguć je direktan pristup datotekama na serveru, pretraživanje po vremenu mjerenja, klasi objekta, njegovoj lokaciji itd.

Za stvaranje seta podataka korišteno je nekoliko tisuća fotografija noćnog neba, u *JPEG* formatu, preuzetih sa SDSS-ovih servera. Svaka fotografija dimenzija je 2048×1489 piksela. Primjer fotografije dan je na slici 4.1.



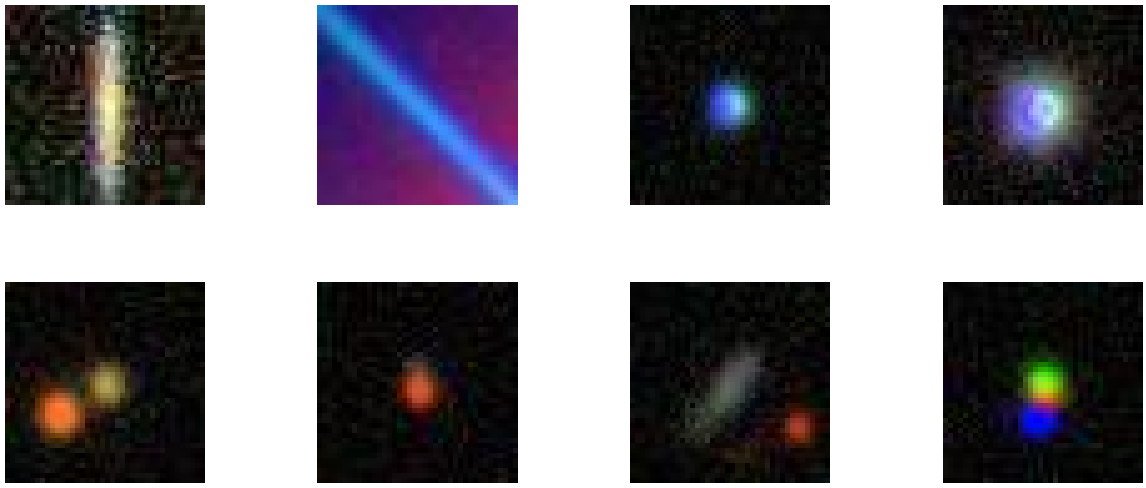
Slika 4.1: Primjer fotografije noćnog neba iz SDSS-III DR12

Korištenje ovakvih fotografija u punoj veličini u radu s neuronskim mrežama bilo bi memorijski iznimno zahtjevno. Iz tog razloga bilo je potrebno izdvojiti pojedinačne objekte s fotografije. Ovakav problem je dobro poznat u polju računalnog vida kao *blob detection* i razvijen je velik broj algoritama s takvom namjenom. Paket algoritama za obradu slika u programskom jeziku Python *scikit-image* [24] sadrži neke

od tih algoritama. Koristio sam `scikit-image.blob_log` algoritam koji se temelji na laplasijanu gausijana (LoG, engl. *Laplacian of Gaussian*). Iako je jedan od sporijih algoritama dostupnih u *scikit-image* paketu, odlučio sam se za njega radi veće točnosti. Kako je ovaj proces trebalo uraditi samo jednom, vremenska komponenta ovog algoritma nije bila toliko bitna.

LoG algoritam detektira samo svijetle maksimume na tamnoj podlozi, tako da je sve fotografije prvo bilo potrebno pretvoriti u crno-bijele. Algoritam kao rezultat daje 2D matricu dimenzija $n \times 3$, gdje je n broj *blobova* pronađenih na slici. Svaki *blob* određen je koordinatama i standardnom devijacijom Gaussianskog kernela σ koji ga je pronašao. Iz standarne devijacije kernela može se izračunati radijus *bloba*, kao $\sqrt{2}\sigma$. Na taj način mogu se eliminirati preveliki objekti na fotografijama, koji vrlo vjerojatno nisu asteroidi. Međutim, zbog očuvanja veće vjerodostojnosti skupa podataka nisam eliminirao objekte po veličini na fotografijama.

Konačno, s koordinatama pojedinačnih objekata na fotografijama, preostalo je svaki objekt izolirati u sličicu dimenzija 40×40 piksela. Nekoliko primjera dobivenih sličica prikazano je na slici 4.2.



Slika 4.2: Primjeri sličica pojedinih objekata sa SDSS-ovih fotografija noćnog neba

Slika 4.2 pokazuje raznovrsnost sličica koje se dobiju iz fotografija u visokoj rezoluciji. Neke sličice sadrže i više objekata, a neke sadrže artefakte različitih uzroka. Kako je naravno cilj da neuronska mreža može detektirati asteroide i na novim slikama, bez čovjekovog uplitanja, artefakti i slične ”problematične” sličice nisu uklanjane iz konačnog skupa podataka.

4.1.1 Veličina

LoG algoritam iz fotografija punih dimenzija, kao što su one na slici 4.1, izdvajao je u prosjeku 120 pojedinačnih objekata. Obrada svih fotografija preuzetih sa SDSS-ovih servera rezultirala je s nekoliko stotina tisuća sličica pojedinačnih objekata. Za treniranje i testiranje neuronske mreže, potrebno je označiti na kojim sličicama se nalazi asteroid, a na kojima ne. Ovo se pokazalo kao veoma zahtjevan zadatak, ponajprije zato što iziskuje iznimno puno vremena. Ako pretpostavimo da iskusnoj osobi treba jedna sekunda da klasificira jednu sličicu, za hipotetskih 300 tisuća sličica potrebno je preko 80 sati rada!

Nadalje, tijekom izgradnje skupa podataka, pokazalo se da po jednoj fotografiji pune rezolucije dolazi svega nekoliko asteroida, na većini fotografija samo jedan. To znači da bi se u konačnom skupu podataka nalazilo manje od 1% sličica asteroida. Očito je da bi bilo veoma lako osmisliti algoritam koji bi bio iznimno uspješan na takvom skupu podataka: mogao bi jednostavno svaku sličicu klasificirati kao ”ne-asteroid” što bi rezultiralo točnošću predviđanja preko 99%.

Zbog takvog nesrazmjera broja asteroida i ostalih objekata bilo je potrebno smanjiti broj sličica objekata koji nisu asteroidi u konačnom skupu podataka. Opasnost toga je naravno smanjenje raznovrsnosti skupa podataka, zbog čega bi se povećala mogućnost da u njemu ne postoji niti jedna sličica neke vrste objekta koja se može pojaviti u kasnijem testiranju i radu neuronske mreže.

Kako bih umanjio mogućnost gubitka neke klase objekata, nasumično sam odabrao sličice iz skupa ”ne-asteroida” tako da njihov broj u konačnom skupu podataka bude bliži broju sličica asteroida. Tijekom testiranja različitih vrsta i verzija neuronskih mreža, skup podataka sam proširivao i nadopunjavao. Naime, znalo se dogoditi da bez obzira na način treniranja, neuronska mreža pogrešno klasificira uvijek iste sličice, iz čega je bilo očito da taj tip objekta nije dovoljno zastupljen u skupu za treniranje.

U konačnici, skup podataka sastojao se od 7132 sličica, od čega je 2820 sličica asteroida te 4312 sličica ostalih objekata.

4.1.2 Podjela

Skup podataka podijelio sam u dva dijela. Konvencionalno, u strojnom učenju, ta dva skupa nazivaju se *skup za treniranje* i *skup za testiranje* i u daljnjem tekstu koristiti ću se tim terminima.

Kao što se da zaključiti iz naziva, skup za treniranje se koristi u fazi treniranja nekog algoritma i obično čini veći dio ukupnog skupa podataka. Skup za testiranje, nasuprot tome, čini manji dio ukupno dostupnog skupa podataka i koristi se za konačno testiranje točnosti algoritma. Uobičajeno je da skup za treniranje čini 80% veličine cijelog skupa podataka, a skup za testiranje preostalih 20%.

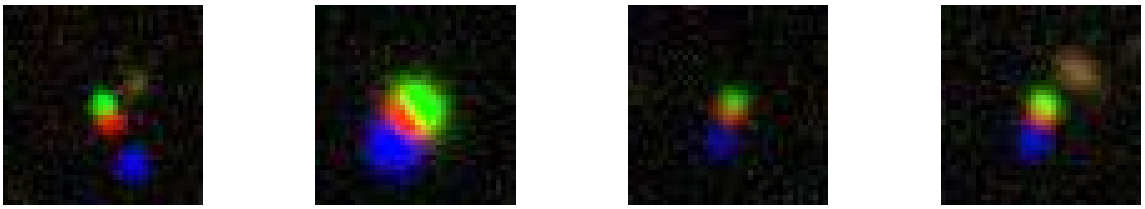
Ključno je osigurati, koliko je to moguće, da oba skupa podataka imaju jednaku distribuciju svih klasa koje nas zanimaju. Ponovno, najočitiji način za to postići je nasumična podjela svih podataka u skup za treniranje i testiranje. Konačna podjela skupa podataka prikazana je u tablici 4.1.

	asteroidi	ostali	
skup za treniranje	2247	3459	
skup za testiranje	563	853	
Σ	**2820**	**4312**	**7132**

Tablica 4.1: Podjela skupa podataka i klasifikacija objekata

S pripremljenim skupom podataka u obliku sličica s (u većini slučajeva) jednim objektom na svakoj, zadatak detekcije se svodi na klasifikaciju u dvije klase: *asteroid* ili *ostalo*. Takvi algoritmi često se nazivaju *binarni klasifikatori*, iz razloga što postoje samo dvije klase za klasifikaciju.

Kao što je već objašnjeno u poglavlju 2.2.2, asteroidi u fotografijama prikupljenim SDSS-ovim teleskopom izgledaju prilično specifično. Još nekoliko primjera dano je na slici 4.3.



Slika 4.3: Primjeri sličica asteroida

Iz samo nekoliko sličica moguće je primjetiti neke pravilnosti u izgledu asteroida. Na sličicama se nalaze tri, donekle odvojena klastera zelene, crvene i plave boje. Ti

klasteri su raspoređeni duž pravca. Već te dvije pravilnosti dovoljne su da čovjek, s vrlo velikom pouzdanošću, može na sličici prepoznati asteroid. Logično je krenuti u tom smjeru: osmisliti algoritam koji će moći detektirati te pravilnosti, te kao i čovjek, moći zaključiti nalazi li se na sličici asteroid ili ne. Druga opcija je u potpunosti izbjeći bilo kakvu analizu podataka i prepustiti algoritmu da pokuša pronaći pravilnosti na sličicama i nekako sâm nauči kako izgleda asteroid na sličici.

U nastavku ću objasniti implementaciju dvaju pristupa i pokušati demonstrirati prednosti i mane:

- klasične neuronske mreže koja kao ulazne podatke prima prethodno analizirane i kvantificirane značajke sličica asteroida
- konvolucijske neuronske mreže koja kao ulazni podatak prima sličicu dimenzija 40×40 piksela

4.2 Klasična neuronska mreža

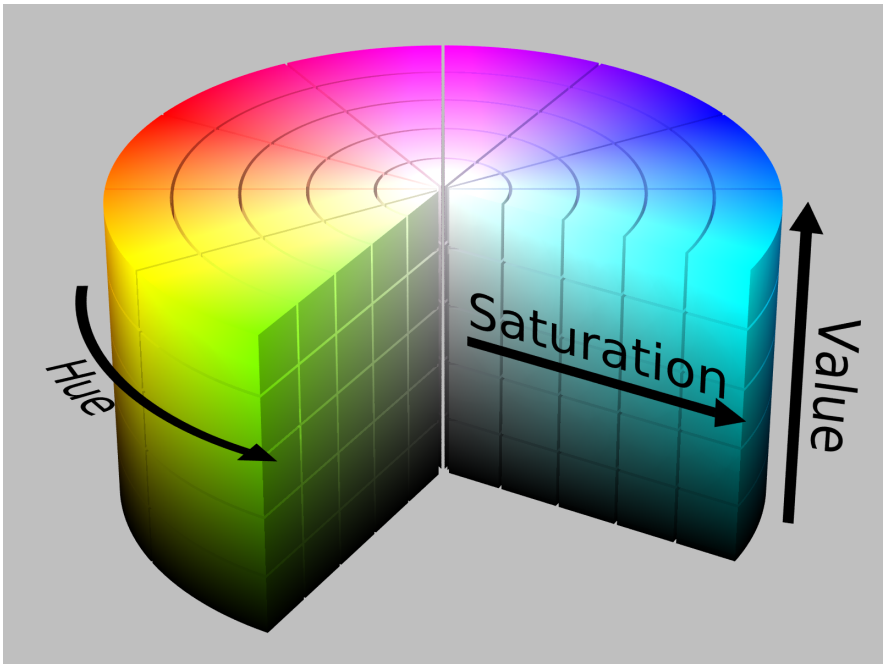
Osnovna ideja za implementaciju klasične neuronske mreže je odlučiti se za nekoliko značajki koje dobro opisuju sličicu asteroida, nekako kvantificirati te značajke i koristiti ih kao ulazne parametre za neuronsku mrežu. U prezentaciji u kojoj je predstavio implementaciju ovakve neuronske mreže [25], Dustin Ingram predložio je tri značajke sličica asteroida:

- omjer nijansi boja
- kolinearnost *klastera* boja
- srednja udaljenost točaka *klastera* od njegovog središta

Odabir ovih značajki u potpunosti se slaže sa slikom 4.3. Sve sličice imaju jasnu raspodjelu boja, vidljivo je da su boje grupirane u tri klastera čija se središta nalaze na istom pravcu te da su zeleni i crveni klaster obično bliže jedan drugome u odnosu na plavi, koji je nešto udaljeniji od njih. Dakle, potrebno je za svaku sličicu nekako izračunati svaku značajku čime bi se dobile tri brojčane vrijednosti koje dobro opisuju izgled asteroida na sličici.

4.2.1 Omjer nijansi boja

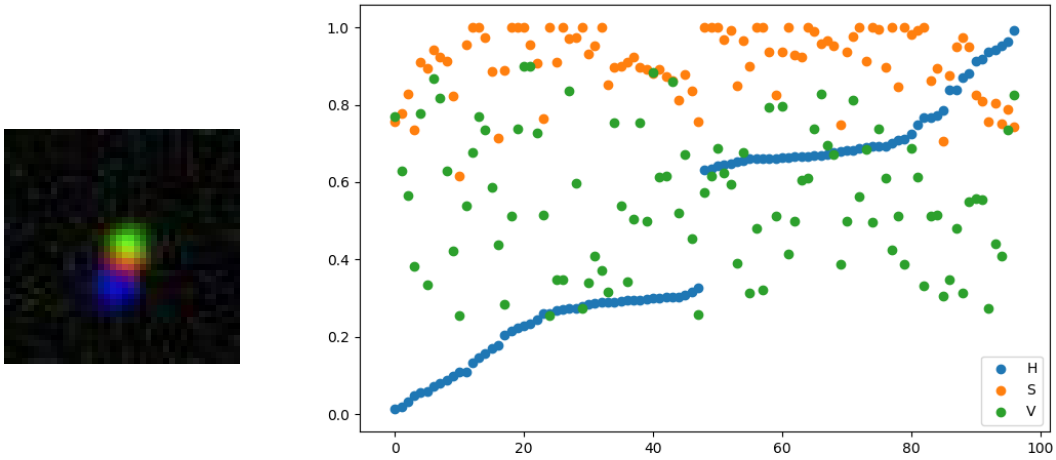
Za izračun, ili bolje rečeno kvantifikaciju, omjera nijansi boja sličice se prvo pretvaraju u HSV [26] sustav boja. HSV stoji za *Hue, Saturation, Value* - nijansa, zasićenje i vrijednost. Kao i u RGB modelu, svaki piksel slike ima tri vrijednosti: u RGB-u su to vrijednosti crvene, zelene i plave boje dok su u HSV-u to nijansa, svijetlina i vrijednost. Grafički prikaz HSV modela nalazi se na slici 4.4.



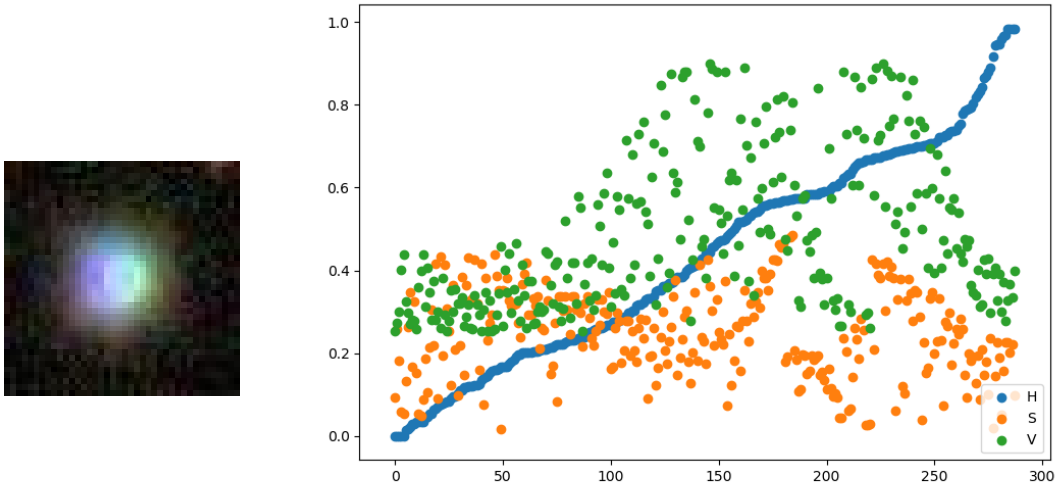
Slika 4.4: Cilindrični prikaz HSV modela boja

Kako znamo koje boje se javljaju, od cijele sličice gledamo samo one piksele koji imaju vrijednosti unutar dvije standardne devijacije od onih nijansi koje očekujemo. Slika 4.5 prikazuje graf HSV vrijednosti sličice asteroida. Vrijednosti su sortirane po nijansama (*hue*), tako da vidimo zastupljenost nijansi u području vrijednosti do oko 0.3 te u području iznad 0.6. Prvi pojas je područje od crvene do zelene, a drugi pojas pripada plavim nijansama. Nasuprotno ovome, slika 4.6 prikazuje graf HSV vrijednosti sličice na kojoj očito nije asteroid. Na sličici su zastupljene sve nijanse boja što se vidi i kao kontinuirana plava linija na HSV dijagramu.

Kao srednje vrijednosti dvaju pojasa nijansi uzete su vrijednosti oko 0.2 za niži pojas i 0.7 za viši pojas nijansi, obje sa standardnim devijacijama 0.03. Zatim se računa *hue_ratio* po formuli 4.1, gdje je a broj piksela nijanse unutar 0.2 ± 0.03, b je broj piksela nijanse unutar 0.7 ± 0.03, a c je broj piksela nijanse izvan ova dva pojasa. Vidi se da će *hue_ratio* biti veći za asteroide, a manji za sličice na kojima nisu



Slika 4.5: HSV dijagram sličice asteroida (pikseli sortirani po H vrijednosti)



Slika 4.6: HSV dijagram sličice ne-asteroida (pikseli sortirani po H vrijednosti)

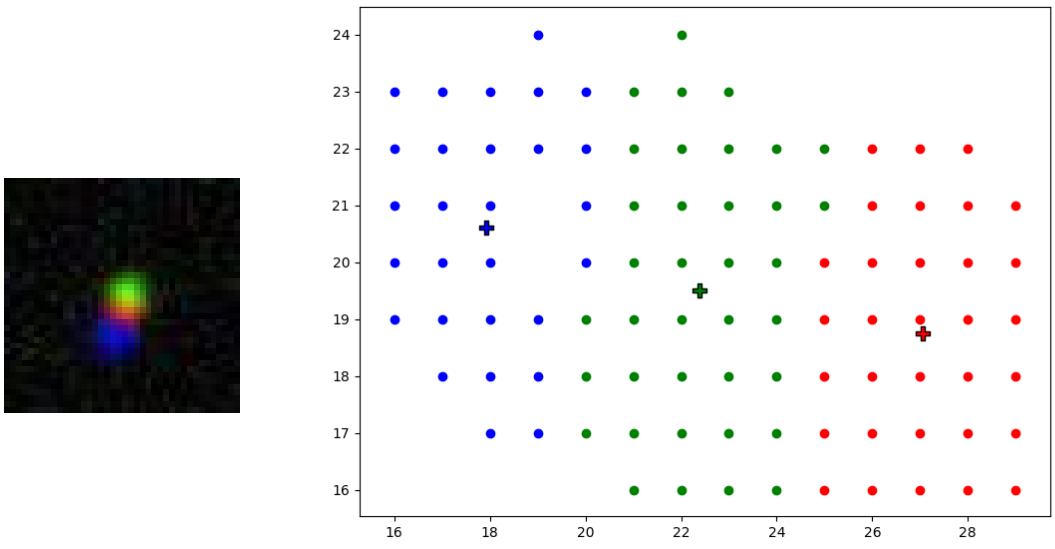
asteroidi.

$$hue\_ratio = \frac{a+b}{a+b+c} \tag{4.1}$$

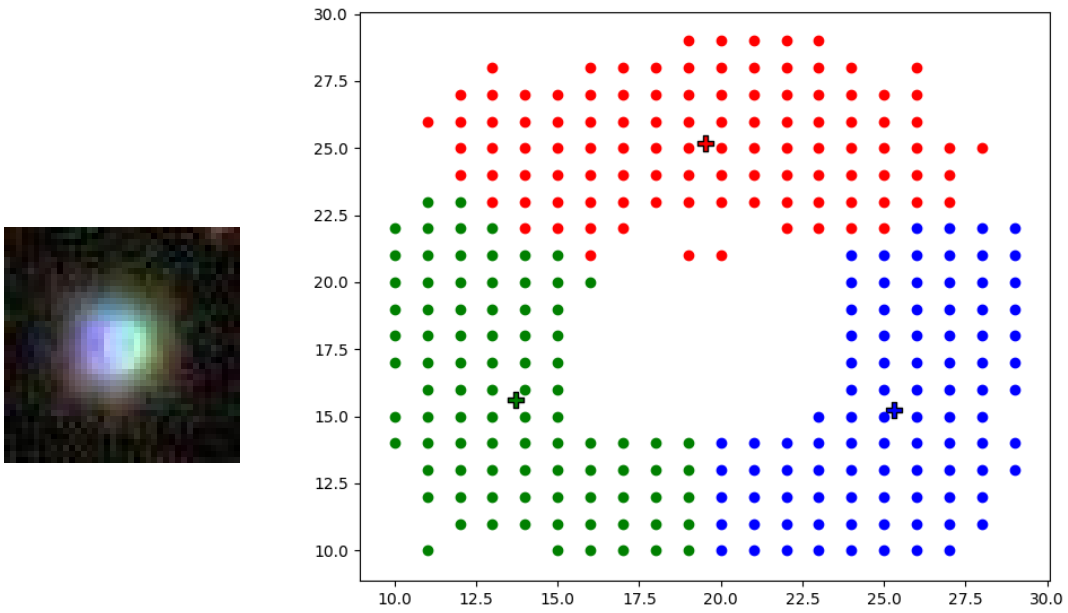
4.2.2 Klasteri boja

Kako bi kvantificirali kolinearnost i udaljenost središta klastera, potrebno je na neki način točke podijeliti u klastere. Za to je korišten *k-means* algoritam iz paketa `scipy` [27]. Taj algoritam nenadziranog strojnog učenja pokušava razvrstati neki skup opažanja u k klastera po zajedničkim obilježjima. U ovom slučaju, zadatak je piksele na slici grupirati u tri klastera po tome koliko su slični. Algoritam radi tako da pokušava minimizirati euklidsku udaljenost između točaka i središta klastera. Očito, točka pripada onom klasteru čijem je središtu najbliža. `kmeans2` algoritam paketa `scipy` direktno vraća koordinate svih centara i listu svih točaka s indeksom centra kojem pripadaju. Iz tih podataka lagano je izračunati kolinearnost središta klastera i srednju udaljenost točaka klastera od njegovog središta.

Slika 4.7 prikazuje rezultate *k-means* algoritma za asteroid, slika 4.8 za ne-asteroid. Vidi se da su u slučaju asteroida točke klastera gušće raspoređene, bliže svom središtu, dok su kod ne-asteroida puno raspršenije. Također, kod asteroida središta klastera su skoro na istom pravcu.



Slika 4.7: Graf klastera dobivenih *k-means* algoritmom za sličicu asteroida



Slika 4.8: Graf klastera dobivenih *k-means* algoritmom za sličicu ne-asteroida

Parametar kolinearnosti odgovara odstupanju od linearnog fita triju središta klastera i za asteroid je puno manja vrijednost nego u slučaju ne-asteroida. Također, za asteroid očekujemo manju srednju udaljenost točaka klastera od njegovog središta.

U tablici 4.2 dani su primjeri ova tri parametra za asteroid i ne-asteroid. Vidimo da je omjer nijansi iz formule 4.1 puno veći za asteroid, dok su vrijednosti kolinearnosti središta i srednje udaljenosti točaka od središta pripadajućeg klastera puno manje.

	omjer nijansi	**kolinearnost središta klastera**	**srednja udaljenost točaka od središta klastera**
asteroid	0.965517241379	0.0101534828808	0.19509791785
ne-asteroid	0.230769230769	0.33251085918	0.531195129269

Tablica 4.2: Primjer vrijednosti parametara za asteroid i ne-asteroid

Naposlijetku, potrebno je ove tri vrijednosti izračunati za svaku sličicu u skupu podataka. Svaka sličica tako postaje jedan vektor x_i oblika 4.2, gdje je zadnja vrijednost vektora, *oznaka*, ili 0 ako na sličici nije asteroid, ili 1 ako je na sličici asteroid. Svaki vektor zapisuje se kao jedan redak u `csv` datoteku. Rezultat ovog procesa su dvije `csv` datoteke, jedna koja sadrži obrađene sličice iz skupa za treniranje i jedna

sličice iz skupa za testiranje. Broj redaka je jednak broju sličica u pripadajućem skupu (tablica 4.1).

$$x_i = \begin{bmatrix} omjer\_nijansi \\ kolinearnost \\ srednja\_udaljenost \\ oznaka \end{bmatrix} \tag{4.2}$$

4.2.3 Implementacija

Za implementaciju neuronske mreže korišten je TensorFlow [28], Python biblioteka otvorenog koda za strojno učenje, a za uvoz i manipulaciju podacima iz `csv` datoteka biblioteka Pandas [29].

Za samu neuronsku mrežu korištena je TensorFlow klasa `DNNClassifier`, kojoj se prosljeđuju ulazni podaci mreže i njena struktura, odnosno broj skrivenih slojeva i neurona u svakom sloju koji je određen varijablom `NODES` u funkciji `build_estimator`:

```
1  def build_estimator(model_dir):
2          # ulazni parametri #
3      hue_rat = tf.contrib.layers.real_valued_column("hue_rat")
4      col_min = tf.contrib.layers.real_valued_column("col_min")
5      dis_min = tf.contrib.layers.real_valued_column("dis_min")
6
7      return tf.contrib.learn.DNNClassifier(
8          model_dir=model_dir,
9          feature_columns=[hue_rat, col_min, dis_min],
10         hidden_units=[ NODES ]
11     )
```

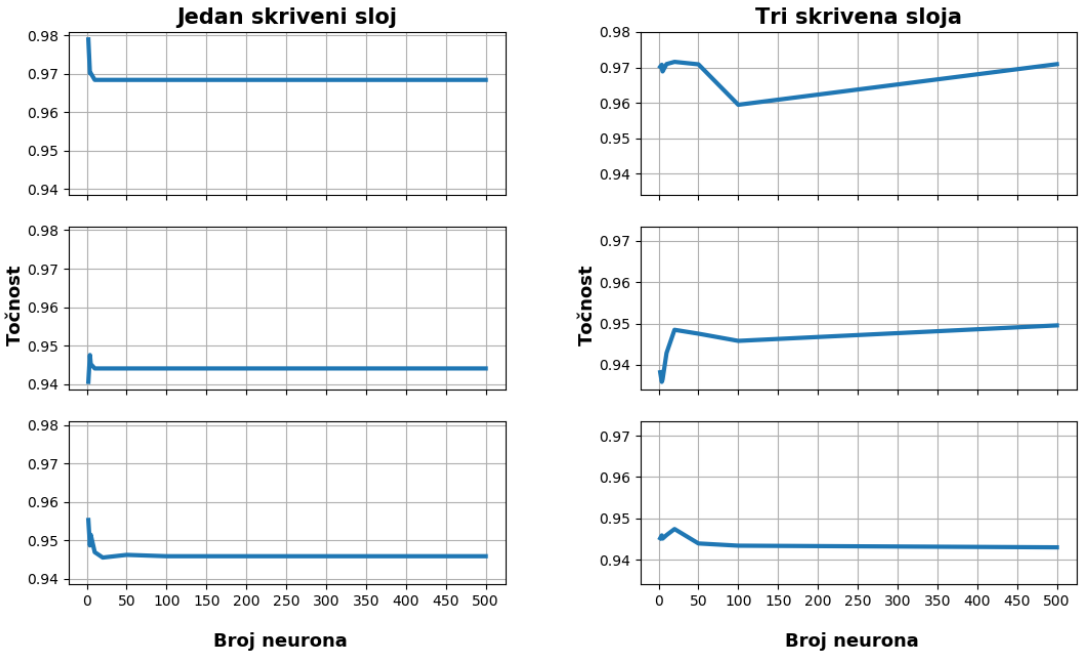
Npr. u slučaju da je `NODES = [100, 100]`, neuronska mreža će imati dva skrivena sloja sa 100 neurona u svakom sloju.

Treniranje neuronske mreže obavlja se pozivom metode `fit` klase `DNNClassifier`. Ova klasa za treniranje koristi *Proximal Adagrad* algoritam [30], koji je nešto drugačiji algoritam od ranije opisanog postupnog opadanja (engl. *gradient descent*). Preostaje samo zadati željeni broj koraka za treniranje mreže i koliko će prolaza izvršiti *backpropagation* algoritam. Naravno da više koraka obično znači i bolju točnost, ali i duže vrijeme treniranja.

4.2.4 Rezultati

Nakon završetka treniranja na odgovarajućem skupu podataka, potrebno je provjeriti točnost neuronske mreže na skupu za testiranje.

Za treniranje i testiranje neuronske mreže korišteno je osobno računalo (Intel Core i5-7200U CPU 2.50GHz s 8 GB radne memorije). Vrijeme treniranja je veoma kratko, ispod deset sekundi i za najveće testirane neuronske mreže.



Slika 4.9: Točnost klasične neuronske mreže s jednim i tri skrivena sloja, za različit (ukupan) broj neurona i na tri skupa podataka [31]

Točnost je testirana za više konfiguracija neuronskih mreža, s jednim, dva i tri skrivena sloja neurona i različitim brojem neurona u svakom skrivenom sloju. U testiranju ove mreže za rad [31] koristili smo skup podataka koji je puno manji od onog koji je analiziran u poglavlju 4.1 i sastoji se od ukupno 2162 sličice (329 asteroida i 1833 ne-asteroida), pokazali smo da najbolje rezultate postiže neuronska mreža sa samo jednim slojem i veoma malim brojem neurona. Slika 4.9 prikazuje postignutu točnost za mreže s jednim i tri skrivena sloja, na tri skupa podataka, od vrha prema dolje, od 200, 400 i 1000 sličica. Na grafovima je na osi apcisa prikazan ukupan broj neurona u mreži. Za mreže s više skrivenih slojeva testirano je više različitih raspodjela neurona po slojevima i uzet je rezultat najbolje raspodjele.

Rezultati pokazuju da mreža s jednim slojem i manje od deset neurona daje go-

tovo konzistentno najbolju točnost, nešto ispod 98%, na sva tri skupa podataka. Mreža s dva skrivena sloja daje skoro identične rezultate kao i ona s tri, na svim skupovima podataka. Pad točnosti na većim skupovima podataka može se objasniti relativno malim skupom podataka.

Ista ova mreža trenirana i testirana na skupu podataka opisanom u poglavlju 4.1 postiže daleko nižu prosječnu točnost od 90.68%, s prosječnim vremenom treniranja 9.4 sekunde.

4.3 Konvolucijska neuronska mreža

Za razliku od klasične, konvolucijska neuronska mreža ne zahtijeva prethodnu obradu podataka, već kao ulazne podatke uzima cijelu sličicu dimenzija 40×40 piksela.

Kada kažem da nije potrebna prethodna obrada podataka, mislim isključivo na obradu i analizu sadržaja same slike. Obrada podataka je svakako potrebna u smislu pripreme podataka za treniranje i testiranje neuronske mreže. U tu svrhu, koristio sam Pythonov `pickle` modul [32] za serijalizaciju objekata. `pickle` algoritam pretvara hijerarhiju objekta u niz bajtova. Analogno, `unpickle` vraća taj niz bajtova natrag u hijerarhijski strukturiran objekt. Ovaj postupak ni u kojem smislu ne mijenja sadržaj ili izgled slike, već samo omogućuje lakše spremanje slika u memoriju, iteriranje po velikom broju slika i njihovo prosljeđivanje neuronskoj mreži.

Same slike su, kao što je već spomenuto, dimenzija 40×40 piksela, svaki piksel opisan je s tri vrijednosti (RGB - crvena, zelena i plava). Dakle, svaka sličica je 3-dimenzionalna matrica oblika (40, 40, 3). Kada se takva matrica razmota, dobiva se vektor duljine 4800. To znači da konvolucijska neuronska mreža u ovom slučaju ima 4800 ulaznih neurona.

4.3.1 Implementacija

Za manipulaciju podacima korištena je Pythonova biblioteka `Scipy` [27], a za neuronsku mrežu TensorFlow [28].

Kao što je spomenuto, neuronska mreža ima 4800 ulaznih neurona. Skriveni slojevi slijede redom: konvolucijski sloj, ”običan” ReLU sloj, konvolucijski, ReLU i naposlijetku izlazni sloj s jednim neuronom. Slojevi sažimanja (engl. *pooling* slojevi) nisu korišteni jer su sličice relativno malih dimenzija.

Mreža je trenirana tzv. *mini-batch* tehnikom, gdje se trenira na malom broju slika, nakon čega se mreža testira i korigira i tako u krug. U tu svrhu, iz skupa za treniranje je izdvojeno 20% sličica za tzv. *validacijski* skup, na kojemu se mreža testirala nakon svakog fiksnog broja *mini-batch*-eva. Struktura ovako modificiranog skupa podataka prikazana je u tablici 4.3.

Moja mreža je trenirana sa 64 sličice po *mini-batch*-u, testirala se na validacijskom skupu svakih 50 *mini-batch*-eva i ukupno prošla 1001 *mini-batch*, što znači da je kroz cijeli (umanjeni) trening skup prošla oko 14 puta.

	asteroidi	ostali	
trening set	1798	2767	
validacijski set	449	692	
test set	563	853	
Σ	**2820**	**4312**	**7132**

Tablica 4.3: Podjela skupa podataka i klasifikacija objekata za konvolucijsku neuronsku mrežu

4.3.2 Rezultati

Za treniranje konvolucijske neuronske mreže na istom osobnom računalu kao i za klasičnu u prosjeku je potrebno 5 minuta. U usporedbi s klasičnom neuronskom mrežom, ovo je 30 puta duže vrijeme treniranja.

Međutim, treniranjem na GPU (*Graphics processing unit*) verziji TensorFlowa, na računalu s GeForce GTX 1060 grafičkom karticom sa 6 GB vlastite memorije i 16 GB radne memorije, prosječno vrijeme treniranja pada na 17.36 sekundi!

Objašnjenje skraćenja vremena treniranja za faktor 17 leži u načinu treniranja neuronskih mreža i rada grafičkih procesora. Kao što je objašnjeno u poglavlju 3.1.3, svaki neuron obavlja prilično jednostavne matematičke operacije, skalarni produkt dvaju vektora i zatim primjenjuje ReLU funkciju na rezultat. Kada vektori postanu jako veliki, broj operacija koje treba obaviti postaje ogroman. Iako su operacije jednostavne, procesor ih mora obaviti jako puno. Obični CPU procesori su dobri u obavljanju jedne komplicirane operacije (odnosno, onoliko operacija koliko procesor ima jezgara), ali su loši u obradi puno uzastopnih operacija. Nasuprot njima, moderne grafičke kartice imaju nekoliko tisuća grafičkih jezgara, koje su više nego dovoljno moćne za obavljati matematičke operacije kakve se pojavljuju u treniranju

neuronskih mreža. Vrijeme testiranja je zanemarivo kratko, ponajviše na sustavu s modernom grafičkom karticom.

Što se tiče točnosti konvolucijske neuronske mreže, ona je u prosjeku iznosila 98.72%.

5 Zaključak

Rezultati koje postižu moderne duboke neuronske mreže na klasifikacijskim zadacima danas graniče s ljudskim učinkom po točnosti, uz neusporedivo kraće vrijeme potrebno za obradu iste količine podataka. Oba pristupa demonstrirana u ovom radu pokazala su visoku točnost na zadatku detekcije asteroida uz više nego zadovoljavajuću brzinu izvođenja.

Klasična neuronska mreža predstavljena u poglavlju 4.2 svojevrstan je pokušaj spoja strojnog učenja i tradicionalnog pristupa programiranju i analizi podataka. Analiza koju radi čovjek, te opisivanje pojave kvantitativnim parametrima i proučavanje njihovih odnosa i korelacija, pouzdana je znanstvena metoda i prirodan pristup rješavanju problema klasifikacije. Ljudi uče na principu prepoznavanja uzoraka te sličnosti i razlika. Zbog toga je logično pokušati naučiti strojeve da tako i sami uče.

Konvolucijska neuronska mreža pak pronalazi, ili bi barem trebala pronaći, te iste uzorke i pravilnosti bez ljudske intervencije. Prethodna analiza strukture i svojstava podataka nije potrebna, što je svakako prednost u problemima s velikim brojem dimenzija, gdje je parametara toliko da čovjek realno ne može proučiti sve njihove skrivene korelacije, ali neuronske mreže možda mogu.

Najveća prednost prvog pristupa je svakako shvatljivost rezultata neuronske mreže. U primjeru asteroida, za bilo koje predviđanje koje neuronska mreža izbaci, lako je pogledati tri parametra čiji je rezultat takvo predviđanje i vidjeti koji je mogući uzrok takvom rezultatu. Npr. možda je preklapanje klastera boja jako naglašeno u specifičnom slučaju zbog čega je *hue_ratio* parametar izvan očekivanih okvira.

Kod konvolucijske neuronske mreže interpretacija načina zaključivanja je puno teža i donekle izvan opsega ovoga rada. Pogreške konvolucijske mreže su uglavnom bili lažni negativi, tj. sličice asteroida za koje je algoritam predvidio da nisu asteroidi. Prvi dojam je da je potrebno proširiti skup za treniranje s više sličica asteroida, što može biti prilično teško, kao što je objašnjeno u poglavlju 4.1.1.

Interpretacija neuronskih mreža obavlja se raznim drugim metodama, proučavanjem težina pojedinih neurona i traženjem grupa neurona koji uzrokuju donošenje određenih zaključaka i slično. Problematika interpretacije rezultata ostaje kao predmet budućeg istraživanja.

Usporedba točnosti dviju neuronskih mreža dana je u tablici 5.1. Konvolucijska

mreža se pokazala kao daleko točnija. Naravno, sama točnost ne može biti dovoljan pokazatelj koliko dobro neki algoritam radi.

	klasična neuronska mreža (4.2)	**konvolucijska neuronska mreža (4.3)**
prosječna točnost (%)	**90.11**	**98.72**

Tablica 5.1: Prosječna točnost neuronskih mreža na skupu podataka iz poglavlja 4.1

U slučaju klasične neuronske mreže nisam išao u detaljno proučavanje pogrešaka i razloga lošije točnosti, iz jednostavnog razloga što se konvolucijska mreža odmah činila kao višestruko bolje rješenje koje bi klasična mreža teško nadvladala čak uz dugotrajan trud oko njenog poboljšanja. Također, taj programski kod je pisan za nešto stariju verziju TensorFlowa, koji se kao paket razvija i mijenja toliko brzo, da bi veće izmjene kôda danas bile prilično mukotrpne.

Najveći problem koji se pokazao pri testiranju konvolucijske neuronske mreže su lažni negativi. Od različitih vrsta mogućih pogrešaka, u problemu detekcije asteroida, lažni negativi su vjerojatno najnepoželjnija vrsta. Puno poželjnija pogreška bi bila više lažnih pozitiva, tj. sličica ne-asteroida koje algoritam klasificira kao asteroide. Takve greške predstavljaju manju opasnost u slučaju stvarne primjene algoritma.

Mogući daljnji koraci u rješavanju ovog problema su, osim proširenja skupa podataka, detaljna analiza neuronske mreže, podešavanje rada konvolucijskih slojeva, dodavanje ili oduzimanje slojeva. Vjerojatnije je dodavanje nego oduzimanje, pošto je implementirana mreža već sada prilično jednostavna. Mišljenja sam da ima dovoljno prostora za napredak.

Problem vremena izvršavanja se pokazao kao vrlo savladiva prepreka. Klasična neuronska mreža na identičnom hardveru će uvijek biti brža od konvolucijske, primarno zbog iznimno jednostavne građe. Međutim, moderne grafičke kartice uklanjaju problem računalne zahtjevnosti dubokih konvolucijskih mreža. Za ovaj rad već jedna grafička kartica je više nego dovoljna za brzo treniranje. Za puno složenije probleme i veće arhitekture sustava strojnog učenja, danas se razvijaju grafičke kartice specifično namijenjene radu neuronskih mreža, kao i sasvim nove vrste čipova [33].

Problem brzine obrade podataka skoro u potpunosti nestaje nakon faze treniranja neuronskih mreža. Klasifikacija novih podataka obavlja se u zanemarivo kratkom vremenu. Naravno, moderan sustav u realnoj primjeni konstantno bi prolazio nove

faze do-treniranja i učenja na vlastitim greškama, za što je i dalje potreban ljudski nadzor i kvalitetna računalna oprema.

Neuronske mreže za detekciju asteroida pokazuju se kao veoma moćan alat koji u kombinaciji s modernom računalnom arhitekturom i pristupom velikim količinama podataka, može odgovoriti na potrebe današnjih astronomskih istraživanja, kako s aspekta brzine obrade enormnih količina podataka, tako i potrebne točnosti klasifikacije.

Dodatak

Dodatak A Obrada slika u Pythonu

Python je, zbog svoje jednostavnosti i svestranosti, danas jedan od najkorištenijih programskih jezika. U nastavi informatike u hrvatskim se gimnazijama u većini slučajeva programiranje uči upravo u Pythonu. Korištenjem specijaliziranih biblioteka moguće je relativno kompleksne radnje izvršiti u nekoliko linija kôda. Jedna od takvih tema je obrada slika. U nastavku ću predložiti ideje o uvođenju ove teme u nastavu u prirodoslovno-matematičkim gimnazijama, ili drugim školama s pojačanim programom nastave informatike.

Koristimo sljedeće Pythonove biblioteke:

- imageio - za učitavanje i spremanje slika
- matplotlib.pyplot - za prikazivanje slika na ekranu
- Numpy i scipy - za osnovnu obradu slika

Za početak će nam trebati prve dvije biblioteke:

```
import imageio
import matplotlib.pyplot as plt
```

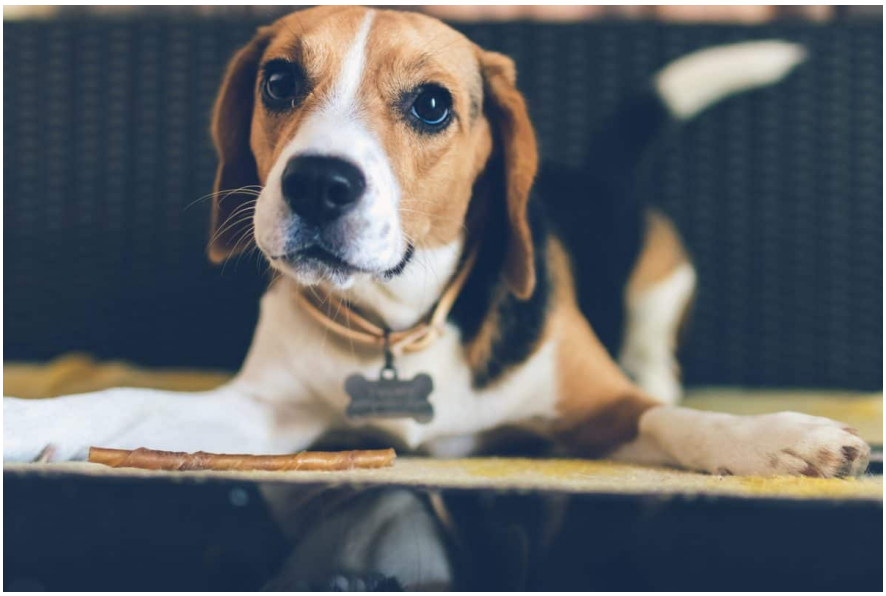
Slika koju ćemo koristiti može se slobodno preuzeti na poveznici `https://goo.gl/t79sHP` i prikazana je na slici A.1. Sliku ćemo spremiti na računalo pod imenom `doggo.jpg`.

Sada ćemo sliku unijeti u naš program funkcijom `imageio.imread()` i spremiti u varijablu `dog`:

```
dog = imageio.imread('doggo.jpg')
```

Možemo provjeriti koji je tip podatka `dog`:

```
type(dog)
```



Slika A.1: Originalna slika

`output: imageio.core.util.Image`

I da se uvjerimo da se zaista radi o slici, prikažimo ju koristeći `pyplot` funkciju `imshow`:

```
plt.imshow(dog) #imshow kreira sliku
plt.show() #prikazuje sliku na zaslonu
```

Na ekranu bi se trebala prikazati slika A.2.

Slika A.2: Slika nakon učitavanja u Python

Ako nas zanimaju dimenzije slike pišemo naredbu:

```
print(dog.shape)
```

output: (800, 1200, 3)

U ovom ispisu prva dva broja označavaju visinu i širinu slike u pikselima, a treći broj znači da se radi o slici u boji, odnosno da je svaki piksel kombinacija tri boje (crvene, zelene i plave). Možemo pogledati što bi dobili ispisom varijable `dog`:

```
print(dog)
```

output: [[[230 215 192] [230 215 192] [230 215 192] ..., [101 100 98] [101 100 98] [101 100 98]] ..., [27 43 66] [27 43 66] [27 43 66]]]

Vidimo da je slika u memoriji zapisana kao lista listi, odnosno kao matrica prirodnih brojeva! Znači da sa slikama možemo raditi kao s listama, tj. matricama. U tu svrhu koristimo biblioteku Numpy i našu sliku pretvaramo u Numpy matricu funkcijom `array`:

```
import numpy as np

dog2 = np.array(dog)
print( type(dog2), dog2.shape )
```

output: <class 'numpy.ndarray'> (800, 1200, 3)

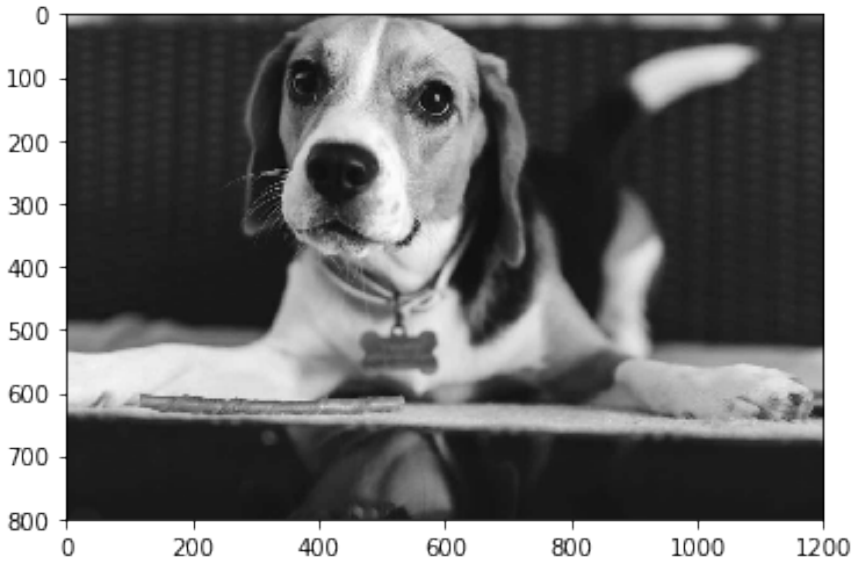
Vidimo da je `dog2` sada objekt klase Numpy matrice, a dimenzije slike su ostale iste.

Što ako želimo crno-bijelu sliku našeg psića? Možemo koristiti modul `scikit-image` biblioteke `scipy` i funkciju `rgb2gray`:

```
from skimage.color import rgb2gray

dog3 = rgb2gray(dog2)
plt.imshow(dog3, cmap = plt.cm.gray)
plt.show()
```

Rezultat je slika A.3. Drugi argument u funkciji `imshow`, `cmap = plt.cm.gray` govori funkciji da je dog3 crno-bijela slika.

Možemo lako izrezati dio slike, samo treba odrediti koji dio slike nas zanima po širini i dužini. Recimo da želimo novu sliku koja će prikazivati glavu našeg psa. Na



Slika A.3: Crno-bijela slika

slici A.3 na koordinatnim osima vidimo da se ona otprilike nalazi po visini od 0 do 500 i po širini od 220 do 800.

```
dog4 = dog3[0:500, 220:800]

plt.imshow(dog4, cmap = plt.cm.gray)
plt.show()
```

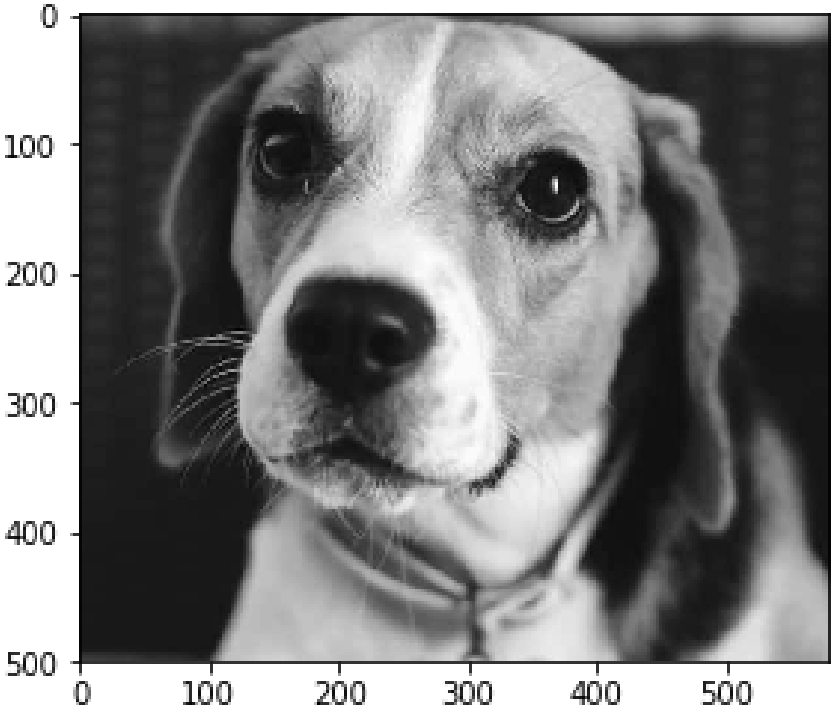
Tako dobijemo sliku A.4. Također, sliku možemo zarotirati gore-dolje funkcijom `np.flipud`, (slika A.5):

```
dog5 = np.flipud(dog4)

plt.imshow(dog5, cmap = plt.cm.gray)
plt.show()
```

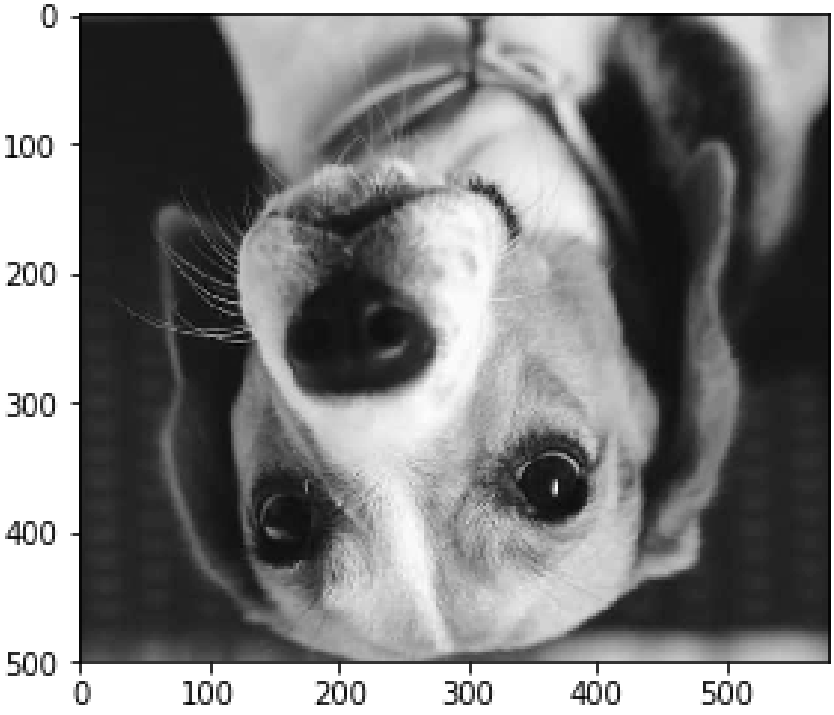
Za lijevo-desno koristimo funkciju `np.fliplr` (slika A.6):

```
dog6 = np.fliplr(dog4)

plt.imshow(dog6, cmap = plt.cm.gray)
plt.show()
```

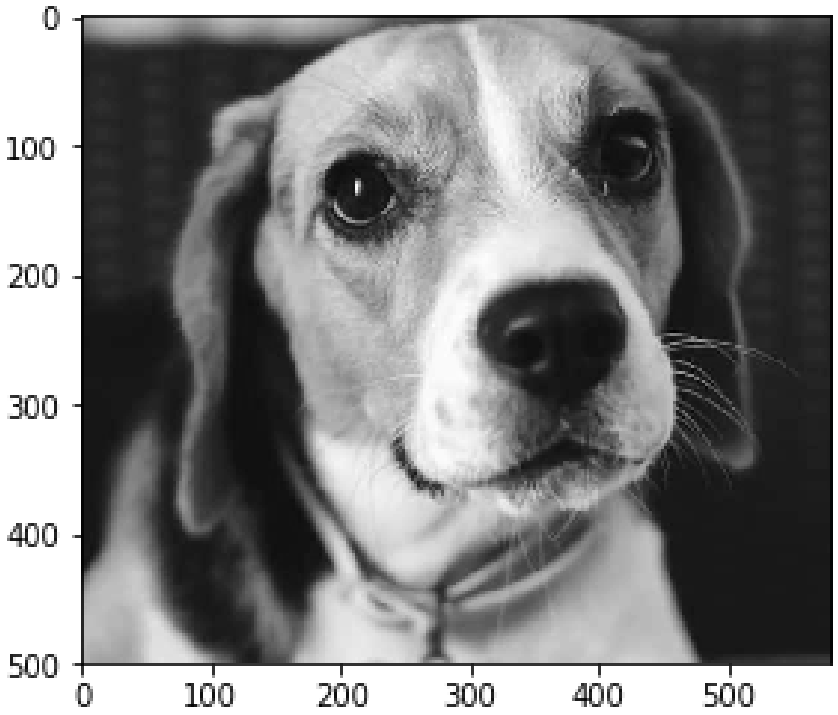
Ovakva primjena programiranja u Pythonu na obradu slika, gdje su rezultati odmah vidljivi i gdje je jasna moguća svakodnevna primjena, može pomoći u povećanju



Slika A.4: Izrezana slika



Slika A.5: Rotacija gore-dolje



Slika A.6: Rotacija lijevo-desno

zainteresiranosti učenika za programiranje. Mladi danas neprestano koriste digitalne kamere na mobitelima i međusobno razmjenjuju slike pa se tema obrade slika može izdvojiti iz možda nekada suhoparnog gradiva informatike u školama.

Nadalje, učenici na veoma vizualan, ali jednostavan način dolaze u kontakt s poznatim i moćnim bibliotekama za numeričku obradu podataka naširoko korištenim u znanstvenoj zajednici. S naprednijim učenicima moguće je dalje istraživati numeričku reprezentaciju slike u memoriji računala i rad s matricama pomoću Numpy biblioteke.

Literatura

[1] *Large Synoptic Survey Telescope*, URL: https://www.lsst.org/, 30. 6. 2018.

[2] *Sloan Digital Sky Survey*, URL: https://www.sdss.org/, 29. 6. 2018.

[3] *International Astronomical Union*, URL: https://www.iau.org/, 28. 6. 2018.

[4] P. Mudrin, *Rock Legends*, Springer Praxis Books, 2016.

[5] Sloan Digital Sky Survey III, *Astronomers Release the Largest Color Image of the Sky Ever Made*, URL: http://www.sdss3.org/press/20110111.largestimage.php, 29. 6. 2018.

[6] Ž. Ivezić i dr., *Statistics, Data Mining and Machine Learning in Astronomy*, Princeton University Press, 2014.

[7] J. E. Gunn i dr., The Sloan Sky Survey Photometric Camera, *The Astronomical Journal* 116 (1998), str. 3040–3081.

[8] *Sloan Digital Sky Survey Moving Object Catalog*, URL: http://faculty.washington.edu/ivezic/sdssmoc/sdssmoc.html, 29. 6. 2018.

[9] Google Resarch Blog, *Assessing Cardiovascular Risk Factors with Computer Vision*, https://research.googleblog.com/2018/02/assessing-cardiovascular-risk-factors.html, 6. 5. 2018.

[10] Eurocontrol, *Predicting flight routes with a Deep Neural Network in the operational Air Traffic Flow and Capacity Management system*, http://www.eurocontrol.int/sites/default/files/publication/files/traffic-prediction-improvements-tpi-factsheet.pdf, 6. 5. 2018.

[11] Google Research Blog, *Predicting flight routes with a Deep Neural Network in the operational Air Traffic Flow and Capacity Management system*, https://research.googleblog.com/2018/03/open-sourcing-hunt-for-exoplanets.html, 6. 5. 2018.

[12] C. Rea i R. S. Granetz, Exploratory Machine Learning Studies for Disruption Prediction Using Large Databases on DIII-D, *Fusion Science and Technology* (2018), str. 1–12.

[13] T. M. Mitchell, *Machine Learning*, New York: McCgraw-Hill, 1997.

[14] F. Rosenblatt, *The Perceptron - a perceiving and recognizing automaton*, teh. izv. 85-460-1, Cornell Aeronautical Laboratory, 1957.

[15] M. Olazaran, A Sociological Study of the Official History of the Perceptrons Controversy, *Social Studies of Science* 26 (1996), str. 611–659.

[16] M. L. Minsky i S. A. Papert, *Perceptrons*, Cambridge: MIT Press, 1969.

[17] D. Silver et al., Mastering the game of Go without human knowledge, *Nature* 550 (2017).

[18] W. McCulloh i W. Pitts, A Logical Calculus of Ideas Immanenet in Nervous Activity, *Bulletin of Mathematical Biology* 5 (1943), str. 115–133.

[19] H. Gray, *Anatomy of the Human Body*, ur. W. H. Lewis, Philadelphia, New York: Lea i Febiger, 1918.

[20] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly, 2017.

[21] D. E. Rumelhart, G. E. Hinton i R. J. Williams, *Leaning Internal Representations by Error Propagation*, ADA164453, Defense Technical Information Center, 1985, 12. 6. 2018.

[22] X. Glorot, A. Bordes i Y. Bengio, *Deep sparse rectifier neural networks*, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, G. Gordon, D. Dunson, i M. Dudík, 2011.

[23] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics* 36 (1980), str. 193–202.

[24] S. van der Walt i dr., *scikit-image: Image processing in Python*, `http://scikit-image.org`, 2014, 25. 6. 2018.

[25] D. Ingram, *Detecting Asteroids with Neural Networks in TensorFlow*, `https://www.youtube.com/watch?v=fj-ZMOi9NrM`, PromptWorks, Philadelphia, 2016, 25. 6. 2018.

[26] A. R. Smith, Color Gamut Transform Pairs, *SIGGRAPH Comput. Graph.* 12 (1978), str. 12–19, URL: `http://doi.acm.org/10.1145/965139.807361`.

[27] E. Jones, T. Oliphant i P. Peterson, *SciPy: Open source scientific tools for Python*, 2001–, URL: `http://www.scipy.org/`, 25. 6. 2018.

[28] Martín Abadi i dr., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015, URL: https://www.tensorflow.org/, 27. 6. 2018.

[29] W. McKinney, *Python for Data Analysis*, O'Reilly Media, 2017.

[30] Y. Singer i J. C Duchi, Efficient Learning using Forward-Backward Splitting, *Advances in Neural Information Processing Systems 22*, ur. Y. Bengio i dr., Curran Associates, Inc., 2009, str. 495–503, URL: http://papers.nips.cc/paper/3793-efficient-learning-using-forward-backward-splitting.pdf.

[31] T. Smolčić, G. Bilalbegović i D. Ingram, *Istraživanje asteroida primjenom neuronskih mreža*, Poster prezentacija, 10. znanstveni sastanak Hrvatskog fizikalnog društva, Baška, 11.-13. listopada 2017.

[32] *pickle - Python object serialization*, https://docs.python.org/2/library/pickle.html, 27. 6. 2018.

[33] Google Cloud Platform Blog, *Google supercharges machine learning tasks with TPU custom chip*, URL: https://cloudplatform.googleblog.com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip.html, 30. 6. 2018.