Klasifikacija oblika galaksija primjenom algoritama u programskom jeziku Python

Bošnjak, Filip

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Filip Bošnjak

KLASIFIKACIJA OBLIKA GALAKSIJA PRIMJENOM ALGORITAMA U PROGRAMSKOM JEZIKU PYTHON

Diplomski rad

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

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

Filip Bošnjak

Diplomski rad

Klasifikacija oblika galaksija primjenom algoritama u programskom jeziku Python

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

Ocjena diplomskog rada: _____

Povjerenstvo: 1._____

2._____

3._____

Datum polaganja: _____

Zagreb, 2022.

Veliku zahvalnost, u prvom redu, dugujem svojoj mentorici Goranki Bilalbegović koja mi je pružila potrebno vodstvo, znanje i ideje za izradu ovog diplomskog rada.
Hvala svim prijateljima i kolegama koji su mi vrijeme studiranje učinili lakšim i lijepšim.
I na kraju, najviše hvala mojim roditeljima, obitelji i mojoj zaručnici zato što su uvijek bili samnom, u sretnim i teškim trenutcima. Bez vas sve ovo ne bi bilo moguće.

Sažetak

Galaksije su gravitacijski povezani skupovi plina, kozmičke prašine, zvijezda i tamne tvari. Vjeruje se da većina velikih galaksija sadrži masivnu crnu rupu u središtu. U svemiru postoje milijuni galaksija od kojih je svaka posebna po svom obliku i veličini. Proučavanje galaksija, njihovog gibanja i transformacije je vrlo značajno za razumjevanje svemira kao cjeline. Današanja teleskopska promatranja neba proizvode ogromnu i sve veću količinu podataka. Ručno označavanje morfoloških karakteristika galaksija je vremenski zahtjevan proces i podložan je pogreškama ljudske prirode. Astrofizičari i programeri pokušavaju riješiti ovaj problem automatizirajući proces klasifikacije i označavanja galaksija. U ovom radu se za analizu pripremljenih slika koriste konvolucijske neuronske mreže. To je najpopularniji algoritam strojnog učenja za analizu i obradu slika te detekciju objekata na njima. Zadatak je računalno razvrstavanje galaksija po obliku na temelju poznatih slika u vidljivom dijelu elektromagnetskog spektra. Ovakvim metodama strojnog učenja je moguće smanjiti pogreške u klasifikaciji galaksija. U metodičkom dijelu diplomskog rada je izložena priprema za održavanje nastavnog sata o algoritmima sortiranja.

Ključne riječi: oblici galaksija, strojno učenje, konvolucijske neuronske mreže, nastava informatike za srednje škole: algoritmi

Galaxies morphology classification using algorithms in programming language Python

Abstract

Galaxies are gravitationally connected systems of gas, cosmic dust, stars and dark matter. It is believed that almost all big galaxies contain a massive black hole in their centers. There are millions of galaxies in the universe, and each of them is special in its shape and size. Study of galaxies, their motion and transformation plays an important role in understanding the universe as a whole. Current telescope observations of the sky produce a huge and growing amount of data. Manual labeling of morphological characteristics of galaxies is time consuming process, prone to human errors. Astrophysicists and programmers are trying to solve this problem by automatising the process of classification and labeling of galaxies. In this diploma thesis convolutional neural networks are used for the analysis of already collected images of galaxies. Convolutional neural networks algorithm is the most popular method in machine learning for the analysis and processing of images and objects detection on them. We describe the classification of shapes for previously known images of galaxies in the visible electromagnetic spectrum. With such methods of machine learning it is possible to reduce errors in the classification of galaxies. In the methodical section we present the preparation material for teaching sorting algorithms in high schools.

Keywords: galaxies morphology, machine learning, convolutional neural networks, teaching computer science in high schools: algorithms

Sadržaj

1 Uvod

Do početka XX. stoljeća čovječanstvu je bila poznata samo jedna galaksija - Mliječna staza. Vjerovalo se da je duga nekoliko stotina svjetlosnih godina i da je to sve što postoji od svemira. Nekoliko drugih galaksija je bilo opaženo teleskopima, ali se vjerovalo da su one samo manji objekti unutar naše galaksije. Edwin Hubble je 1923. godine usmjerio svoj teleskop u blijede mrlje svijetla na tamnom noćnom nebu [1]. Koristio je Hookerov teleskop na planini Wilson u Kaliforniji. To je bio najveći i najjači teleskop tog vremena sa zrcalom promjera 250 centimetara. Hubble je njime po prvi put razlučio pojedinačne zvijezde u svjetlosnoj mrlji koju danas zovemo Andromeda. Hubble je do 1929. godine potpuno preoblikovao način na koji razumijemo svemir. Pokazao je da svemir ne samo sadrži milijune galaksija, nego i da se širi. Hubble je također ustanovio shemu za klasifikaciju galaksija. Podijelio je galaksije na eliptične i spiralne, a nakon toga na više podvrsta. Tako je Hubble postavio temelje moderne klasifikacije galaksija.

Galaksije nam pokazuju kako je tvar u svemiru raspoređena na velikim skalama. Kako bismo bolje razumjeli povijest i narav svemira potrebno je poznavati trenutni raspored tvari i pratiti evoluciju od početka postojanja svemira. Mnoga pitanja još uvijek ostaju neodgovorena: npr. kako su se galaksije formirale, odakle tolika raznolikost među oblicima i veličinama galaksija te koji je odnos galaksije i crne rupe u njenom središtu. Na putu do odgovora na ta pitanja može nam pomoći klasifikacija galaksija [2].

Cilj ovog rada je analiza algoritma strojnog učenja koji klasificira oblike galaksija po njihovim slikama u vidljivom dijelu elektromagnetskog spektra. Razmatramo tri oblika galaksija: eliptične, spiralne i lećaste (između eliptičnih i spiralnih). U drugom poglavlju predstavit ćemo galaksije i njihove oblike. U trećem poglavlju izložene su osnove strojnog učenja i opisane su konvolucijske neuronske mreže koje se koriste u ovom radu. U četvrtom poglavlju analiziraju se rezultati i predstavljaju programerski alati te proces obrade podataka. Koriste se konvolucijske neuronske mreže za konstrukciju modela koji predviđa oblik galaksija. Model se trenira tako što uči iz velikog broja slika ručno klasificiranih galaksija. U metodičkom poglavlju predstavljena je priprema za nastavni sat iz informatike koji se bavi algoritmima sortiranja. Prvo su opisani jednostavni algoritmi sortiranja kao što su sortiranje metodom mjehurića

(engl. *bubble sort*) i sortiranje izborom najmanjeg elementa (engl. *selection sort*). Nakon toga se prelazi na kompleksnije algoritme: sortiranje sjedinjavanjem (engl. *merge sort*) i brzo sortiranje (engl. *quick sort*).

2 Galaksije

2.1 O galaksijama

Galaksija je gravitacijski povezan sustav kozmičke prašine, plina, tamne tvari i velikog broja zvijezda [3, 4]. Riječ galaksija je izvedena od grčke riječi $\gamma\alpha\lambda\alpha\xi\iota\alpha\zeta$ što znači "mliječni" (referenca na Mliječnu stazu). Veličine galaksija su različite, od patuljastih koje sadrže nekoliko stotina milijuna zvijezda (10^8) do divovskih galaksija kojima pripadaju stotine bilijuna zvijezda (10^{14}). Promjer većine galaksija je između 1000 i 100 000 parseka, tj. između 3000 do 300 000 svjetlosnih godina. Galaksije su gravitacijski organizirane u grupe, klastere i superklastere. Golemi svemirski prostor sadrži puno galaksija. Ti prelijepi i tajanstveni svemirski sustavi su, ne samo mjesta rođenja i evolucije zvijezda, nego su i svemirski svjetionici koji nam omogućavaju proučavanje svemira na ogromnoj kozmološkoj skali. Većina pitanja koja se tiču postanka, razvoja i oblikovanja galaksija ostaju neodgovorena. Ta pitanja nastavljaju biti važan aspekt kozmologije koja se bavi proučavanjem strukture i evolucije svemira kao cjeline.

Vremenske skale nastanka i evolucije galaksija neusporedivo su veće od trajanja života ljudi. U svemiru imamo na raspolaganju milijune i milijune galaksija koje su na različitim udaljenostima od nas. A kako svjetlosti treba vremena da stigne do Zemlje s daleke galaksije, možemo promatrati slične galaksije u različitim vremenskim fazama kroz povijest svemira. Promatranje puno galaksija na sve većim i većim udaljenostima je kao promatranje galaksija u sve mlađem i mlađem svemiru. Galaksije nam nisu samo zanimljve individualno, već također igraju važnu ulogu u proučavanju svemira kao cjeline. One su tragovi postanka i razvoja svemira.

Galaksije su samo dio svemira, u kojem dominiraju tamna tvar i tamna energija. Proučavanje galaksija i njihovog odnosa s tamnom energijom dovelo je do jednog od najvećih otkrića 20. stoljeća - svemir se širi. Galaksije se ubrzano udaljavaju od Zemlje i to zato što se prostor-vrijeme koje je između njih širi. To je posljedica izmjerenih pomaka prema crvenom (engl. *redshift*) valnih duljina koje dolaze s udaljenih galaksija. Mjereni pomak prema crvenom je veći ako je galaksija udaljenija. Edwin Hubble je 1929. godine na temelju 22 mjerenja bližih galaksija zaključio da je brzina

v_r kojom se galaksije udaljuju proporcionalna njihovoj udaljenosti d od nas [3]:

$$v_r \approx H_0 d \tag{2.1}$$

Nešto kasnije potvrđena je ova Hubbleova relacija i izmjeren je parametar H_0 koji danas zovemo Hubbleova konstanta. Ona iznosi između 60 i 75 $kms^{-1}Mpc^{-1}$. Ako su prosječne brzine galaksija bile približno konstantne, one bi bile sve u istoj točki u vremenu t_H prije sadašnjosti:

$$t_H = \frac{1}{H_0} = \frac{1}{2,37 \cdot 10^{-18} s^{-1}}, \tag{2.2}$$

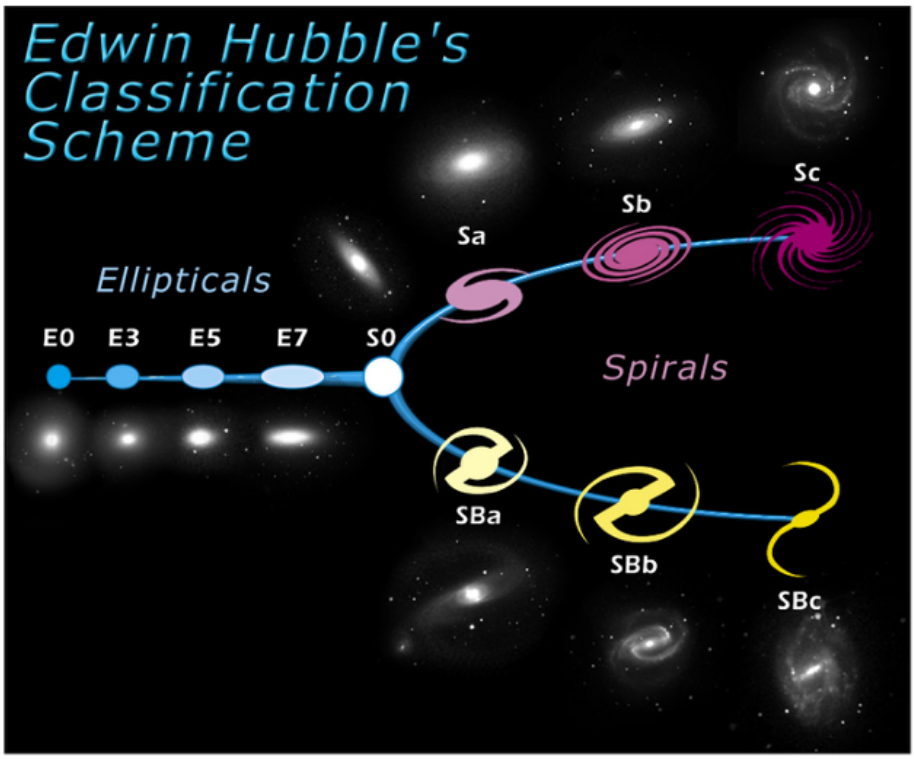
što iznosi oko 14 milijardi godina. To je gruba mjera starosti svemira koju smo dobili direktno iz proučavanja evolucije galaksija i njihovog odnosa s tamnom tvari. Mnogo drugih informacija je skriveno u galaksijama i svemiru. To sve čini istraživanja galaksija, njihovog postanka i razvoja vrlo važnim za kozmologiju i fiziku.

2.2 Oblici galaksija

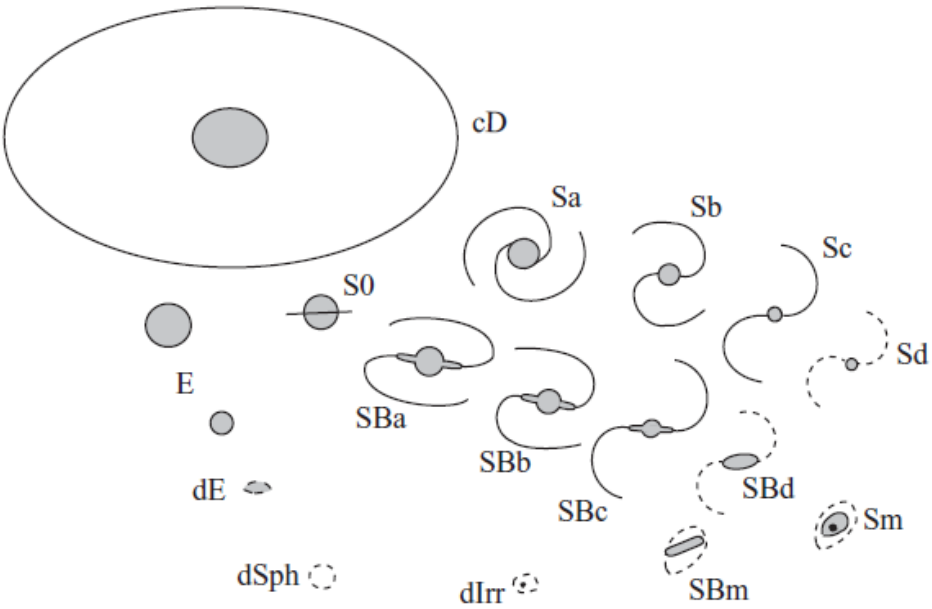
Oblici galaksija koje vidimo na Zemlji su uglavnom rezultat elektromagnetskog zračenja koje proizvode njihove zvijezde [3, 5]. Galaksije koje možemo opisati poznatim oblicima su spiralne i eliptične. Eliptične galaksije su blago izravnati elipsoidni sustavi, a spiralne galaksije imaju spljošten disk. Ako razmatramo povijest svemira, eliptične i spiralne galaksije su rane i kasne galaksije, respektivno. Galaksije su često mješavina ova dva oblika. To čini klasifikaciju oblika galaksija izazovnom.

Edwin Hubble je 1926. godine razvio shemu klasifikacije galaksija prema obliku koja je još uvijek u širokoj upotrebi (Slika 2.1). Hubbleova sekvenca ima raspon od ranih elipsoida do kasnih spiralnih galaksija. Hubble je prepoznao tri glavna tipa galaksija: eliptične, lećaste i spiralne. Dodao je još jednu kategoriju nepravilnih galaksija za one galaksije koje je nemoguće smjestiti u jednu od tri navedene kategorije. Detaljniji oblik Hubbleove sheme klasifikacija oblika galaksija prikazan je na Slici 2.2.

Eliptične galaksije su okruglaste i glatke [3]. One nemaju dovoljno hladnog plina i zbog toga imaju mali broj mladih plavih zvijezda. Eliptične galaksije su najčešće grupirane u velika jata. Najveće od njih, cD galaksije na modificiranoj Hubbleovoj shemi (Slika 2.2), se nalaze u najgušćim dijelovima jata, vrlo su velike (stotine kps) i imaju luminozitet i do stotinu puta veći od Mliječne staze. Primjer eliptične galaksije



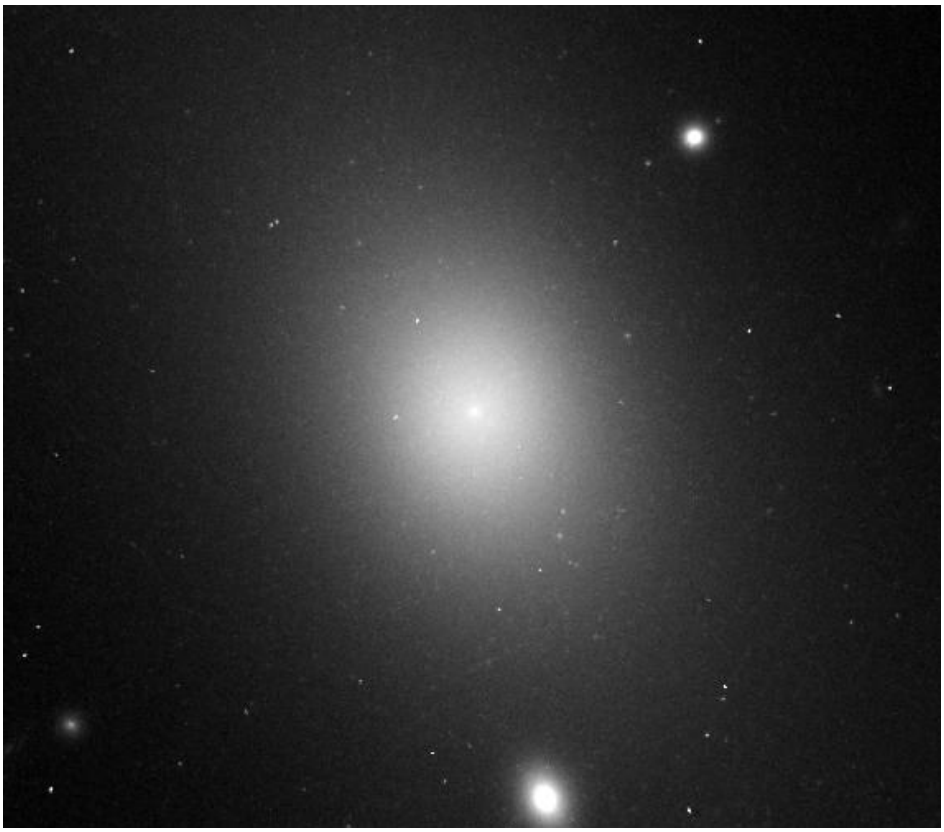
Slika 2.1: Hubbleova klasifikacija oblika galaksija [6]



Slika 2.2: Modificirana Hubbleova klasifikacija oblika galaksija [3]

je prikazan na Slici 2.3.

Lećaste (engl. *lenticular*) galaksije imaju rotirajući disk i centralno ispupčenje (engl. *bulge*) [3]. Na Hubbleovoj shemi ih označavamo sa S0 i one su u području tranzicije galaksija iz spiralnih u eliptične. S eliptičnim galaksijama dijele nedostatak

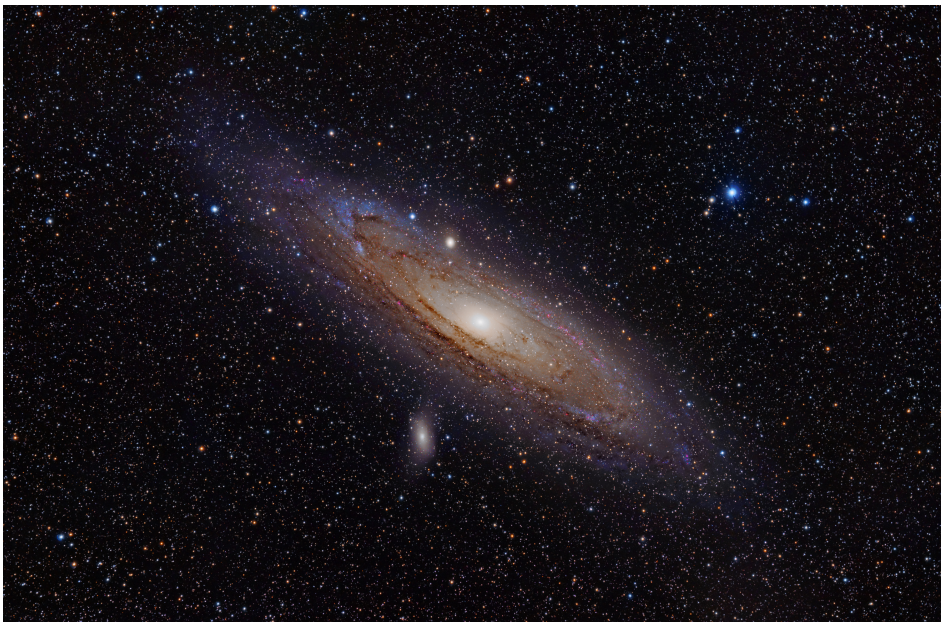


Slika 2.3: Eliptična galaksija IC 1101. To je jedna od najvećih poznatih galaksija [7]

kozmičke prašine te smiještanje unutar prostora koji je gusto popunjen galaksijama. Kao spiralne galaksije, imaju tanak, rotirajući zvijezdani disk. Za razliku od spiralnih galaksija, disk lećastih galaksija ne proizvodi aktivno nove zvijezde. Primjer lećaste galaksije prikazan je na Slici 2.4.

Slika 2.4: NGC 5866, lećasta galaksija u zviježdu Zmaj [8]

Spiralne galaksije (Slika 2.5) dobile su ime po sjajnim spiralnim krakovima osobito bogatim plavim svijetlom koje je bilo vrlo lako snimiti ranim fotografskim pločama [3]. Spiralne se galaksije sastoje od središnjeg područja s velikom koncentracijom zvijezda. To područje se još zove i ispupčenje (engl. *bulge*). Oko središnjeg ispupčenja zvijezde formiraju spiralnu strukturu te oblikuju spljošteni disk. Na Hubbleovoj shemi spiralne galaksije označene su slovom S. Sa SB označene su galaksije kojima iz središta izvire ravna šipkasta struktura koju tvore zvijezde. Mala slova (a, b, c,...) pokazuju strukturu spiralnih krakova. Naša galaksija, Mliječna staza, klasificirana je prema obliku kao Sc ili SBc. Te oznake su prisutne u modificiranoj Hubbleovoj klasifikaciji (Slika 2.2).



Slika 2.5: Primjer spiralne galaksije: nama najbliža galaksija Andromeda (NGC 224) [9]

2.3 Opažanje galaksija

Teleskopi kojima promatramo svemir opažaju galaksije na svim valnim duljinama elektromagnetskog spektra. Postoje radio, infracrveni (IR), optički, ultraljubičasti (UV) teleskopi, te oni koji koriste X-zračenje i gama-zračanje. Teleskope možemo postaviti na Zemlji, ali i slati u svemir. Možemo detaljno opažati jednu galaksiju, ili određenu površinu neba gdje se nalaze milijuni galaksija. Promatranjem površine neba nastaju pregledi neba. Jedan od najpoznatijih je Sloanov digitalni pregled neba (engl. *Sloan Digital Sky Survey, SDSS*) [10].

Neki od teleskopa koji se bave opažanjem svemira u području radiovalova su GMRT (engl. *Giant Meterwave Radio Telescope*), ATCA (engl. *Australia Telescope Compact Array*), Karl G. Jansky VLA (engl. *Very Large Array*), LOFAR (engl. *Low Frequency Array*) i ALMA (engl. *Atacama Large Millimeter Array*) [11, 12].

Zemljina atmosfera propušta vrlo malo infracrvene svjetlosti. Ako želimo dektektirati IR zračenja svemirskih objekata, teleskope moramo lansirati van atmosfere Zemlje. Najpoznatiji IR teleskopi su: Spitzer, Herschel i WISE [11, 13]. NASA je 25. 12. 2021. lansirala James Webb teleskop koji će, između ostalog, promatrati galaksije u IR podrčju [2].

Jedan od najvećih i najnaprednijih optičkih teleskopa na Zemlji je VLT (engl. *Very Large Telescope* [11, 14]). Izgrađen je u pustinji Atacama u sjevernom Čileu. Sastoji

se od četiri zasebna teleskopa čija zrcala imaju promjer od 8,2 m. Teleskopi mogu raditi zajedno i tvore veliki teleskop poznat kao ESO VLT interferometar. On može stvarati fotografije koje su i do 25 puta detaljnije nego fotografije napravljene svakim teleskopom pojedinačno.

Promatranje svemira sa Zemlje je zahtjevan posao jer je Zemljina atmosfera vrlo promjenjiva i često zamršena. Zato je svemir puno efikasnije promatrati iz svemira. Hubbleov teleskop je prvi veliki optički teleskop koji smo poslali u Zemljinu orbitu [11, 15]. Lansiran je u svemir 1990. godine. Nalazi se nisko u Zemljinoj orbiti, na visini od 547 km. Opaža svjetlost elektromagnetskog spektra od ultraljubičastog, preko vidljivog pa sve do bliskog IR.

Teleskop Chandra (engl. *Candra X-ray observatory*) aktivno opaža rendgensko zračenje iz svemira [11, 16]. Nalazi se visoko u Zemljinoj orbiti na 139 000 km. X-zrake potječu iz vrlo energetski nabijenih dijelova svemira, kao što su supernove (eksplozije zvijezda), jata galaksija i tvari koje orbitiraju oko crnih rupa. Najpoznatiji teleskopi koji detektiraju gama zračenje su: INTEGRAL (engl. *INTErnational Gamma-Ray Astrophysics Laboratory*), Fermi i MAGIC (engl. *Major Atmospheric Gamma Imaging Cherenkov*) [17].

3 Osnove strojnog učenja

3.1 O strojnom učenju

Strojno učenje je dio područja računalnih znanosti koje se zove umjetna inteligencija [18–20]. Definira se kao algoritam koji na temelju prijašnjeg iskustva postaje sve bolji u predviđanju ishoda, a bez da je za to eksplicitno programiran. Najvažnija podjela strojnog učenja je na nadzirano i nenadzirano.

U nadziranom strojnom učenju algoritam uči iz ulaz/izlaz parova podataka [19]. Za određeni skup podataka poznato je što algoritam za svaki ulaz mora dati na izlazu. Podaci koje koristimo u strojnom učenju imaju neke osobine koje ih opisuju, tj. obilježja (značajke, engl. *features*). Jedan od primjera je algoritam koji razlikuje sliku psa od slike mačke. Algoritmu dajemo slike pasa s oznakom ”pas” i slike mačaka s oznakom ”mačka”. Algoritm na temelju tih parova podataka uči i nakon toga može predvidjeti rezultat za neoznačene slike. Dvije glavne kategorije nenadziranih algoritama su klasifikacija i regresija [21].

Kod nenadziranog strojnog učenja podaci su neoznačeni i algoritam nema ”učitelja” [19]. Iz prirode problema poznato je koja nas rješenja zanimaju. Algoritam sam nalazi obrasce u podacima koji su potencijalno korisni. Primjer je otkrivanje glavne teme u velikom skupu blog postova, ili otkrivanje DDoS napada koji preplavljuje server ogromnom količinom zahtjeva. Najpoznatiji tip nenadziranih algoritama je grupiranje (engl. *clustering*).

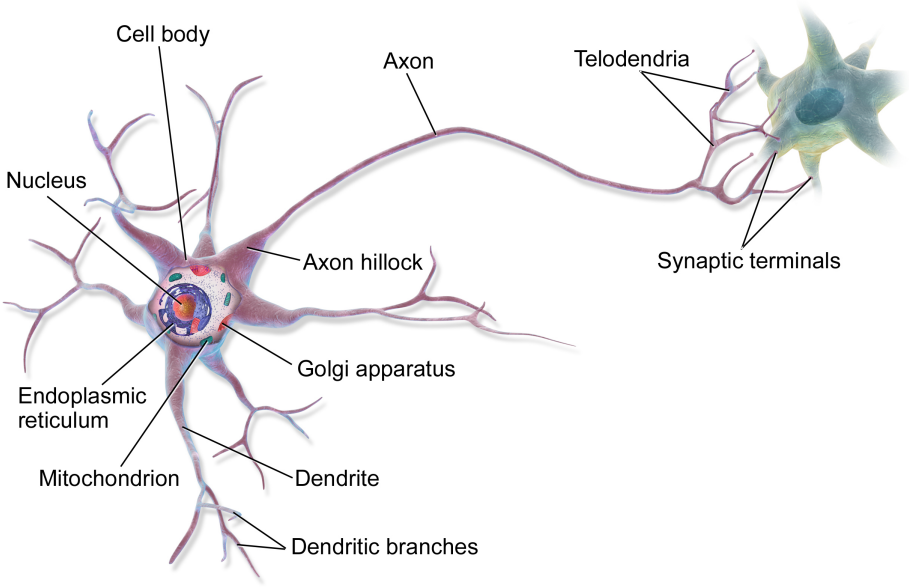
Strojno učenje ima važnu ulogu u algoritmima računalnog vida, obradi zvuka, fotografija i videa, medicini, samonavodećim autima i robotici. Algoritmi strojnog učenja analiziraju medicinsku dijagnostiku pacijenta, reakcije korisnika na objave društvenih mreža, pad i rast cijena dionica na tržištu, čestice u visokoenergetskom ubrzivaču te znanstvene fotografije, kao sto su slike galaksija i drugih astronomskih objekata.

3.2 Neuronske mreže

3.2.1 Biološki neuroni

Prije uvođenja umjetnih neurona pogledajmo u kratkim crtama kako funkcionira biološki neuron (Slika 3.1). Neuron je temeljna jedinica ljudskog i životinjskog mozga.

Maleni dio mozga veličine zrna riže sadrži preko 10 000 neurona. Svaki neuron u prosjeku ostvaruje vezu sa 6000 drugih neurona i na taj način je ostvarena ogromna biološka mreža koja nam omogućava da analiziramo svijet oko nas i riješavamo kompleksne probleme s kojima se susrećemo. Neuron je optimiziran za primanje informacija od susjednih neurona, procesiranje tih informacija i slanje dobivenih rezultata drugim stanicama.



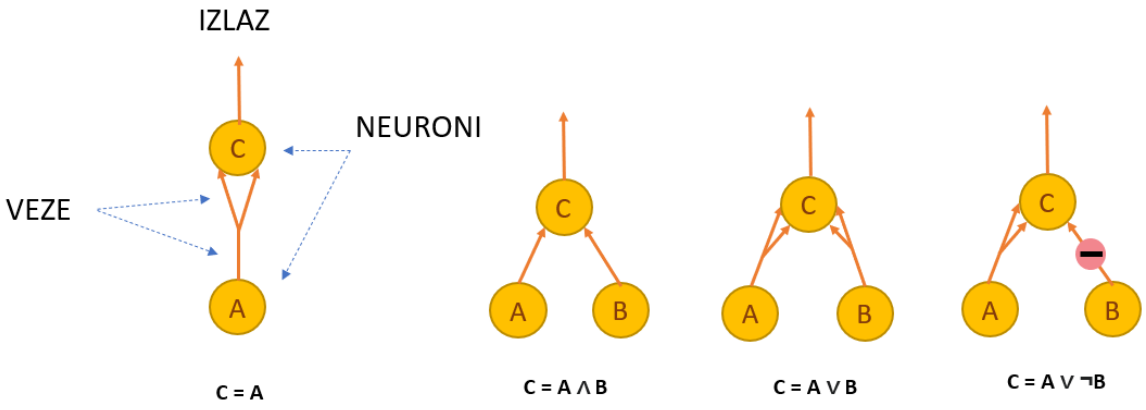
Slika 3.1: Anatomija multipolarnog neurona koji sadrži više dendrita [22]

Neuron prima ulazne impulse preko dentrita, razgranatih struktura koje su produžetak staničnog tijela neurona. Svaka od ovih ulaznih veza dinamički jača i slabi u ovisnosti o tome koliko je korištena. Mozak ljudskog bića uči ojačavanjem veza među neuronima u neuronskim mrežama. Jačina svake veze određuje doprinos pojedinih ulaznih signala izlaznom signalu.

3.2.2 Umjetni neuroni

Umjetne neuronske mreže (engl. *Artificial Neural Network* ili ANN, skraćeno *Neural Network* ili NN) se sastoje od čvorova (engl. *nodes*). Čvorovi su umjetna, matematička reprezentacija biološkog neurona. Među prvim i najjednostavnijim modelima bila je neuronska mreža koju su 1943. godine predložili Warren McCulloch i Walter Pitts [19]. Njihova mreža je imala jedan ili više binarnih ulaza i jedan binarni izlaz. Ti ulazi i izlazi funkcionirali su kao on/off sklopke. Umjetni neuron bi slao aktivacijski izlazni signal kada bi određen broj njegovih ulaznih signala bio aktivan.

McCulloch i Pitts su pokazali da se čak i s ovakvim primitivnim modelom može sagraditi mreža umjetnih neurona koja računa bilo koji logički izračun. Pogledajmo to na nekoliko primjera (Slika 3.2).



Slika 3.2: Jednostavna neuronska mreža za logičko računanje

- Prva mreža na Slici 3.2 (s lijeva na desno) je jednostavna funkcija identiteta. Ako A neuron pošalje aktivacijski signal onda će se C neuron aktivirati i proslijediti svoj aktivacijski signal dalje. Ako je neuron A neaktivan, onda će i neuron C ostati neaktivan.

- Druga mreža računa logičku operaciju I (engl. *AND*). Samo ako i neuron A i neuron B pošalju aktivacijski signal, onda i neuron C šalje aktivacijski signal. Bilo koja druga kombinacija (aktiviran samo A ili B, ili niti jedan) rezultira u neaktiviranom neuronu C.

- Treća mreža računa logičko ILI (engl. *OR*). Za aktivaciju neurona C dovoljno je da ili A, ili B (ili oba) pošalju aktivacijski signal. Ako niti A, niti B nije aktiviran, neuron C ne šalje aktivacijski signal.

- U četvrtoj neuronskoj mreži računa se logičko NE (engl. *NOT*). Ovdje je C neuron aktivan samo ako je neuron A aktiviran, a neuron B inhibiran. Kao i kod biološkog neurona, umjetni neuroni mogu biti dizajnirani tako da ulazni signal inhibira izlazni signal i gasi neuron.

Od ovakvih jednostavnih mreža mogu se sastaviti kompleksnije mreže i sustavi mreža.

3.2.3 Perceptron

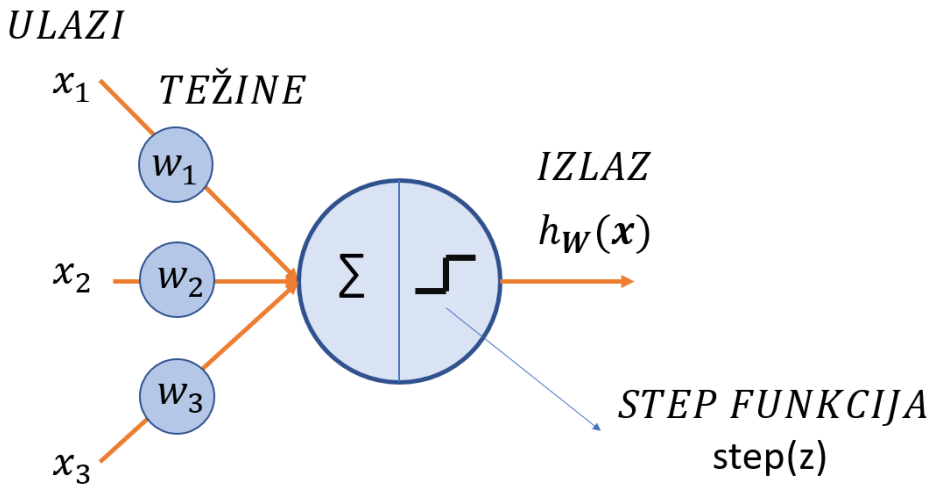
Perceptron je struktura neuronskih mreža koju je predložio Frank Rosenblatt 1957. godine [23]. Perceptroni su kompleksniji od mreža s binarnim ulazom i izlazom: njihovi ulazi i izlazi nisu on/off vrijednosti, već brojevi [19]. Svakom ulaznom neuronu $(x_1, ..., x_n)$ pridružuje se težinski parametar ili težina (engl. *weight*). Perceptron računa sumu svih ulaza pomnoženih sa pripadnom težinom:

$$z = w_1 x_1 + w_2 x_2 + ... + w_n x_n = \mathbf{x}^T \mathbf{w} \tag{3.1}$$

Tada na tu sumu primjeni step funkciju i vraća rezultat:

$$h_w(\mathbf{x}) = step(z), \qquad z = \mathbf{x}^T \mathbf{w} \tag{3.2}$$

Perceptron je prikazan na Slici 3.3.



Slika 3.3: Shema perceptrona

Step funkcija $f : \mathbb{R} \longrightarrow \mathbb{R}$ je definirana kao:

$$f(x) = \sum_{i=0}^{n} \alpha_i \chi_{A_i}(x), \qquad \forall x, \alpha_i, n \geq 0 \in \mathbb{R} \tag{3.3}$$

U formuli 3.3 A_i su intervali, a χ_A je karakteristična funkcija od A definirana kao:

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \tag{3.4}$$

Primjeri step funkcija koje se koriste u perceptronima su:

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{3.5}$$

i

$$f(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{3.6}$$

Mreža uči tako da podešava težine na vezama. Učenje podešavanjem težina još zovemo treniranjem mreže i modela kao cjeline. U ulazni sloj perceptrona dodajemo još jedan parametar koji nazivamo pristranost (engl. *bias*). Za slučaj binarne klasifikacije je $b = 1$. Primjenom linearne algebre možemo računati izlaze perceptrona. Računamo to koristeći jednadžbu:

$$h_{\mathbf{W},\mathbf{b}}(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b}) \tag{3.7}$$

- $\mathbf{X}$ predstavlja matricu ulaznih vrijednosti.
- Matrica težina $\mathbf{W}$ sadrži težinske parametre.
- Vektor pristranosti $\mathbf{b}$ sadrži odgovarajuće parametere. Parametri pristranosti služe da održe učenje aktivnim i u slučaju slabih signala. Ti parametri su također podložni promjenama tokom treniranja modela tj. učenja.
- ϕ predstavlja aktivacijsku funkciju. Ona ima zadnju riječ u tome hoće li se neuron aktivirati ili neće. Primjeri aktivacijskih funkcija su opisani u 3.2.4.

Algoritam treniranja perceptrona uveo je Rosenblatt, a inspiraciju je našao u *Hebbovom pravilu*. Donald Hebb, kanadski psiholog i vrlo utjecajan na području neurologije i neuroznanosti, uočio je da kada jedan biološki neuron u ljudskom mozgu aktivira drugi, često se dogodi da veza između ta dva neurona ojača. Ovo je kasnije formulirano u Hebbovo pravilo koje kaže da se težina veze između dva neurona povećava kada god oni imaju isti izlaz. Perceptroni su trenirani na principu ovog pravila u koje uvodimo i pogrešku koju je mreža napravila u predviđanju izlaza. Perceptronu dajemo ulazne podatke, a on nam vraća predviđanje. Za svako

krivo predviđanje, ojačavaju se veze koje bi za taj slučaj davale točna predviđanja. Podešavanje težina računa se prema jednadžbi:

$$w_{i,j} = w_{i,j} + \eta(y_i - \hat{y}_j)x_i \tag{3.8}$$

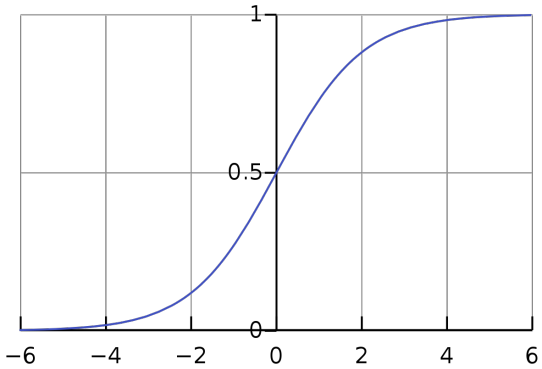
- $w_{i,j}$ ovdje označava težinu veze između i-tog ulaza i j-tog izlaza.
- x_i je i-ta ulazna vrijednost
- y_i predstavlja ciljani izlaz j-tog neurona
- $\hat{y}_j$ je stvarni, trenutni izlaz j-tog neurona
- η je brzina kojom model uči

3.2.4 Aktivacijske funkcije

Jednostavne aktivacijske funkcije su step funkcije koje su definirane u jednažbama 3.3-3.6. Jedna od aktivacijskih funkcija koja je povijesno značajna za strojno učenje je *sigmoid* funkcija. Jednadžba te funkcije je:

$$\phi(x) = \frac{1}{1 + e^{-z}} \tag{3.9}$$

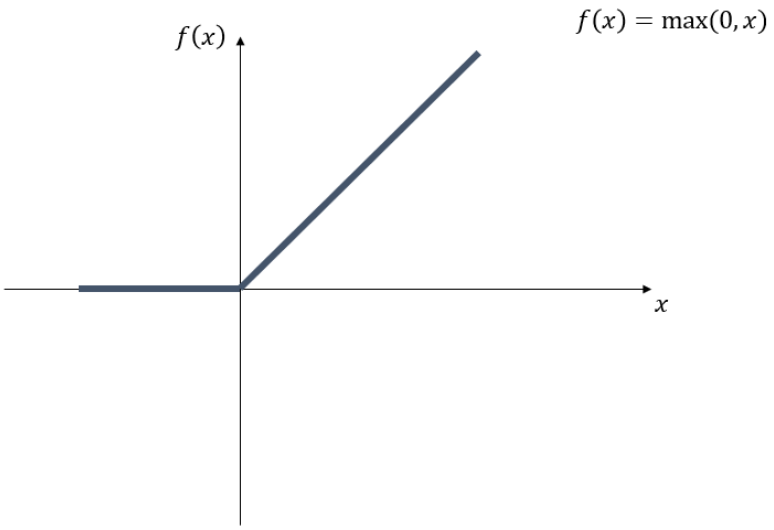
Ova funkcija daje uvijek vrijednosti u intervalu $(0, 1)$ što je važno za algoritme binarne klasifikacije. Sigmoid funkcija je prikazana na Slici 3.4.



Slika 3.4: Sigmoid funkcija [24]

Sigmoid aktivacijska funkcija se sve rijeđe koristi jer je dokazano da se prilikom treniranja neuronske mreže gube informacije na jedinicama sa sigmoid funkcijom.

Zglobnica ili ispravljena linearna jedinica (engl. *rectified linear unit, ReLU*) je nastala po uzoru na biološke neurone. Ako je ulazna vrijednost nula ili manja od nule, rezultat funkcije je nula. Ako je ulazna vrijednost bilo koji broj veći od nule, izlazna vrijednost je taj broj. ReLU aktivacijska funkcija je prikazana na Slici 3.5.

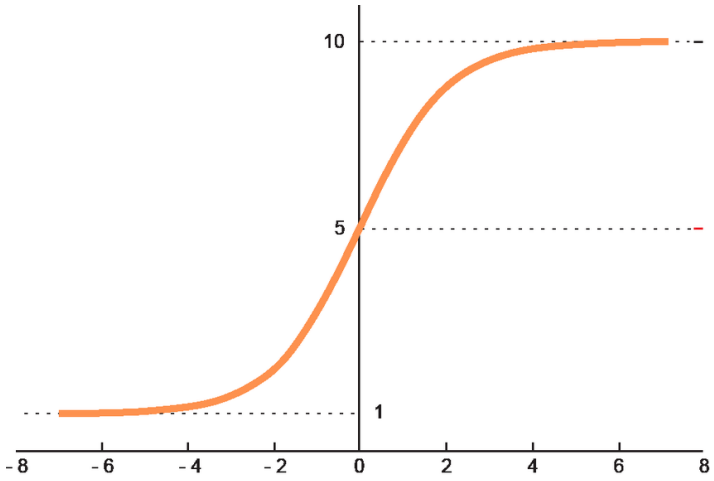


Slika 3.5: ReLU aktivacijska funkcija

Softmax aktivacijska funkcija se najčešće koristi kada imamo više od dvije, tj. N mogućih izlaznih kategorija [25]. Softmax, ili normalizirana eksponencijalna funkcija, je matematička funkcija koja pretvara vektor brojeva u vektor vjerojatnosti, gdje su vjerojatnosti proporcionalne relativnoj skali svake vrijednosti u vektoru brojeva. Neka imamo N izlaznih vrijednosti na N dimenzionalnom realnom vektorskom prostoru. Definiramo softmax funkciju $\sigma : \mathbb{R}^K \longrightarrow [0,1]^K$

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}, i = 1, ..., K \tag{3.10}$$

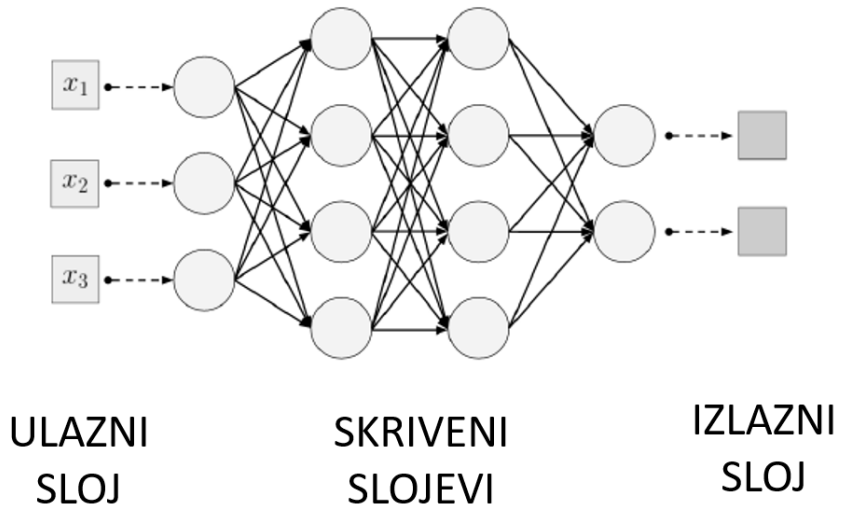
Prikazana je na Slici 3.6. Softmax je složeniji oblik argmax aktivacijske funkcije. Argmax vraća index najvećeg elementa u vektoru. Softmax skalira vrijednosti u vektoru tako da se svaka vrijednost nalazi između 0 i 1. Ovo postižemo tako da za svaku vrijednost izračunamo omjer te eksponencirane vrijednosti i sume svih eksponenciranih vrijednosti, kao što je prikazano u jednadžbi 3.10.



Slika 3.6: Softmax aktivacijska funkcija

3.2.5 Višeslojne neuronske mreže

Danas se u primjenama strojnog učenja najčešće koriste višeslojne neuronske mreže [19]. Takve mreže sadrže jedan ulazni i jedan izlazni sloj neurona te jedan ili više skrivenih slojeva (Slika 3.7). Dubina modela određena je brojem slojeva. Što više skrivenih slojeva mreža ima to je dublja, odakle i dolazi naziv duboko učenje i duboke neuronske mreže. U realnim modelima su najčešći potpuno povezani slojevi (engl. *fully connected layers, FCL*). U FC slojevima je svaki neuron povezan s neuronima susjednog sloja. Svaki neuron u određenom sloju najčešće koristi istu aktivacijsku funkciju. Ulazni parametri neurona u skrivenom sloju su izlazi neurona iz prethodnog sloja. Svaki od slojeva oblikuje podatke koji se kreću kroz mrežu.



Slika 3.7: Potpuno povezana višeslojna neuronska mreža [20].

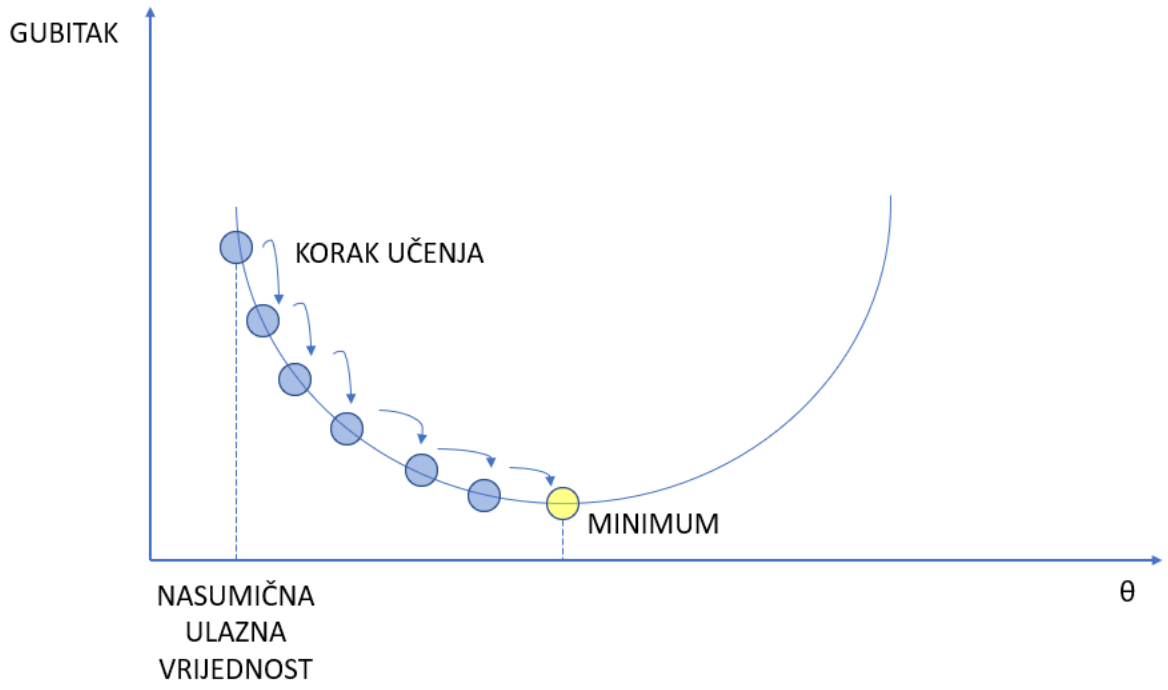
Postupci treniranja višeslojnih perceptrona su dugo istraživani. David Rumelhart, Geoffrey Hinton i Ronald Williams 1986. godine objavljuju članak [26] koji je bio ogroman napredak na području strojnog učenja. Uvodi se algoritam treniranja koji se zove propagacija unatrag. Ovaj je algoritam u širokoj upotrebi još i danas, a temelji se na metodi gradijentnog spusta. Proces učenja bilo kojeg algoritma koji se zasniva na težinama je proces podešavanja tih težina kojim određeni bitovi informacije dobivaju veći značaj, a nekim bitovima se značaj smanjuje. Neuronska mreža uči odnose i uzorke tako što radi nasumično nagađanje koje se bazira na ulazima i mjeri koliko je predviđanje točno. Algoritam optimizacije nagrađuje određen dio mreže ili cijelu mrežu za dobro predviđanje, a kažnjava ju za loše predviđanje. Kako bismo detaljnije razumjeli učenje neuronskih mreža, moramo objasniti algoritme na kojima se ono temelji, a to su gradijentni spust i propagacija unatrag.

3.2.6 Gradijentni spust

Gradijentni spust je iterativni algoritam optimizacije koji traži lokalni minimum diferencijabilne funkcije [19, 27]. Algoritam je univerzalan, ima široku upotrebu i koristi se za rješavanje raznih problema u mnogim područjima znanosti. Neka je zadan testni skup označenih podataka. Funkcija gubitka (engl. *loss function*) mjeri odstupanje izlaznog rezultata, tj. predviđene oznake i stvarne oznake testnog uzorka. To radi za sve uzorke u testnom skupu. Ulazne podatke i oznake ne možemo mijenjati. Mreža mora podesiti težine kako bi davala precizna predviđanja i smanjila funkciju gubitka.

Algoritam gradijentnog spusta uzima nasumičnu točku funkcije gubitka, izračuna lokalni gradijent prema određenom parametru θ i u sljedećem koraku kreće u smjeru najstrmijeg spusta. Jednom kada gradijent postaje jednak nuli, stigli smo do lokalnog minimuma (Slika 3.8). Važan parametar gradijentnog spusta je veličina koraka koju ćemo zvati brzinom učenja (engl. *learning rate*). Ako je korak učenja premali, algoritmu će trebati velik broj iteracija kako bi došao do lokalnog minimuma. Ovo čini naš algoritam sporim i neefikasnim. S druge strane, ako nam je korak prevelik, algoritam će preskočiti preko minimuma na drugu stranu grafa. Ovako dolazi do velikih promašaja i do mogućnosti divergencije algoritma. Od velike je važnosti naći ispravan korak učenja.

Želimo naći minimum funkcije $C(x)$, gdje x može biti $x = x_1, x_2, x_3, ...$ [28]. Ograničimo se na samo dvije varijable $C(x_1, x_2)$. Za jednostavne funkcije dvije varija-



Slika 3.8: Shematski prikaza traženja minimuma gradijentnim spustom

ble često možemo odrediti lokalni minimum ako nacrtamo tu funkciju, ili izračunamo analitički njenu prvu i drugu derivaciju. Međutim, u dubokom učenju radimo s funkcijama mnogo varijabli koje mogu imati mnogo lokalnih minimuma. Traženje minimuma izračunavanjem prvih i drugih derivacija ovdje ne pomaže. Pogledajmo što se događa ako se pomaknemo za mali korak Δx_1 u x_1 smjeru te Δx_2 u x_2 smjeru. Znamo da tada vrijedi:

$$\Delta C \approx \frac{\partial C}{\partial x_1}\Delta x_1 + \frac{\partial C}{\partial x_2}\Delta x_2 \tag{3.11}$$

Želimo naći Δx_1 i Δx_2 takve da ΔC bude negativan jer to znači da se krećemo prema lokalnom minimumu. Definiramo vektor promjene $\Delta \mathbf{x} \equiv (\Delta x_1, \Delta x_2)$. Također definiramo *gradijent* od C kao:

$$\nabla C \approx \left(\frac{\partial C}{\partial x_1}, \frac{\partial C}{\partial x_2}\right) \tag{3.12}$$

Sada možemo jednadžbu za ΔC (3.11) napisati kao:

$$\Delta C \approx \nabla C \cdot \Delta x \tag{3.13}$$

Ova jednadžba nam može reći kako da izaberemo Δx tako da je ukupan ∇C uvijek

negativan. Ako izaberemo Δx kao:

$$\Delta x = -\eta \nabla C \tag{3.14}$$

onda je η parametar koji predstavlja brzinu učenja. Iz jednadžbe 3.13 se vidi da vrijedi:

$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta |\nabla C|^2 \tag{3.15}$$

Znamo da vrijedi $|\nabla C|^2 \geq 0$. To nam garantira da je $\Delta C \leq 0$, što povlači stalan pad funkcije C ako variramo x prema jednadžbi 3.14. Ovo je način na koji želimo da algoritam gradijentnog spusta funkcionira.

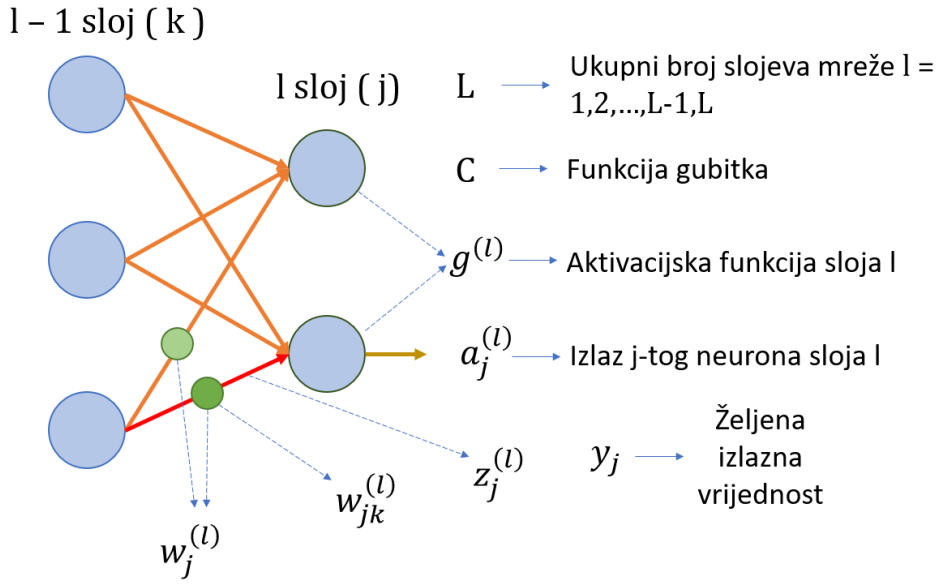
3.2.7 Propagacija unatrag

Algoritam propagacije unatrag je vrlo važan za rad neuronskih mreža. Rumelhart, Hinton i Williams su definirali da propagacija unatrag "uzastopno podešava težine unutar mreže kako bi se minimizirala razlika između stvarnih i željenih izlaznih vektora" [26]. Propagacija unatrag ima za cilj minimizirati funkciju gubitka podešavanjem težina i pomaka. Količinu podešavanja određuje gradijent funkcije gubitka po težinama i pomacima [20, 29, 30].

Neka je mreža odradila računanje za jedan ulazni vektor i dobila neki izlazni vektor. Taj izlazni vektor, tj. predviđanje mreže, uspoređujemo sa stvarnom vrijednosti koju želimo dobiti. Ovdje koristimo funkciju gubitka koja nam vraća mjeru pogreške predviđanja. Tada algoritam računa koliko koji neuron u izlaznom sloju doprinosi pogreški. Ovo određuje primjenjujući pravilo ulančanog deriviranja koje čini ovaj korak brzim i preciznim. Algoritam tada mjeri koliko je pojedina veza iz prethodnog sloja u mreži doprinjela pogreški, također koristeći pravilo ulančanog deriviranja, sve dok ne dođe do ulaznog sloja. Računamo gradijent greške svih veza u mreži propagirajući je natrag kroz mrežu (odakle dolazi i naziv algoritma). Na kraju gradijentni spust izvodi mala podešavanja svih težina veza u mreži.

Pogledajmo kako to izgleda matematički [29]. Neka imamo L slojeva u našoj neuronskoj mreži koji su indeksirani kao $l = 1, 2, 3, ..., L-1, L$. Neka su za sloj l koji promatramo pojedini neuroni indeksirani kao $j = 0, 1, 2, ..., n-1$, a neuroni u prethodnom sloju $l-1$ indeksirani kao $k = 0, 1, 2, ..., n-1$. Označavat ćemo željenu

izlaznu vrijednost j-tog neurona u izlaznom L sloju kao y_j, a ulaznu vrijednost kao $z_j^{(l)}$. Funkciju gubitka ćemo označavati kao C. Težinu koja povezuje neuron k u sloju $l-1$ s neuronom j u sloju l označavamao kao $w_{jk}^{(l)}$, a vektor j-tog neurona koji sadrži sve veze s prethodnim slojem $l-1$ kao $w_j^{(l)}$. Aktivacijsku funkciju sloja l označavat ćemo s $g^{(l)}$, a aktivacijski izlaz j-tog neurona u sloju l kao $a_j^{(l)}$. Postupak je shematski prikazan na Slici 3.9.



Slika 3.9: Shematski prikaz notacije za algoritam propagacije unatrag

Uočimo da ako oduzmemo izlaznu vrijednost j-tog neurona iz sloja l i željenu izlaznu vrijednost y_j, tu razliku kvadriramo i sumiramo po svim neuronima sloja l dobivamo funkciju gubitka [29]:

$$C = \sum_{j=0}^{n-1} (a_j^L - y_j)^2 \tag{3.16}$$

Također uočimo važnu stvar o ulaznoj vrijednosti neurona j. Znamo da je ta vrijednost suma pojedinih izlaznih vrijednosti prethodnog $l-1$ sloja izražena kao umnožak $w_{jk}^{(l)} a_k^{(l-1)}$. Stoga ukupna ulazna vrijednost neurona j je jednaka sumi po svim neuronima prethodnog sloja $l-1$ koji označavamo s k te imamo:

$$z_j^{(l)} = \sum_{k=0}^{n-1} w_{jk}^{(l)} a_k^{l-1} \tag{3.17}$$

Nadalje, primjetimo da je izlazna vrijednost j-tog neurona koju računa aktivacijska

funkcija izlaznog L sloja ovisi o ulaznoj vrijednosti $z_j^{(l)}$. Možemo pisati:

$$a_j^{(l)} = g^{(l)}(z_j^{(l)}) \tag{3.18}$$

Također znamo da težina spoja između j-tog neurona $w_j^{(l)}$ i neurona iz prethodnog sloja sudjeluje u određivanju ulazne vrijednosti $z_j^{(l)}$ te možemo pisati:

$$z_j^{(l)} = z_j^{(l)}(w_j^{(l)}) \tag{3.19}$$

Sada možemo izraziti funkciju gubitka C kao kompoziciju funkcija kombinirajući jednadžbe 3.16 - 3.18. Iz jednadžbe 3.16 se vidi da pojedini član sume C_j ima oblik:

$$C_j = (a_j^L - y_j)^2 \tag{3.20}$$

Možemo utvrditi da je C_j funkcija od a_j^L jer je y_j konstanta. Pišemo:

$$C_j = C_j(a_j^L) \tag{3.21}$$

Kombinirajući jednadžbe 3.20, 3.18 i 3.19 dobivamo:

$$C_j = C_j(a_j^{(L)}(z_j^{(L)}(w_j^{(L)}))) \tag{3.22}$$

Ova observacija će nam biti od velike koristi kod deriviranja kompozicije funkcija koristeći pravilo ulančanog deriviranja.

Cilj algoritma propagacije unatrag je podesiti težine veza u mreži tako da funkcija gubitka bude minimalna. Način na koji treba podesiti težine algoritam može jednostavno izračunati uzimajući parcijalnu derivaciju funkcije gubitka po pojedinoj težini u mreži [30]:

$$\frac{\partial C}{\partial w_{jk}^{(L)}} \tag{3.23}$$

Ispitujemo kako mala varijacija težine w_{jk} utječe na funkciju gubitka. Uzimajući jednadžbu 3.21 u obzir imamo:

$$\frac{\partial C}{\partial w_{jk}^{(L)}} = \left(\frac{\partial C}{\partial a_j^{(L)}}\right)\left(\frac{\partial a_j^{(L)}}{\partial z_j^{(L)}}\right)\left(\frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}}\right) \tag{3.24}$$

Kada parcijalno deriviramo svaki od tri člana u jednadžb i3.24, dobivamo:

$$\frac{\partial C}{\partial w_{jk}^{(L)}} = 2(a_j^{(L)} - y_j)(g'^{(L)}(z_j^L))(a_k^{(L-1)}) \tag{3.25}$$

Ova jednadžba nam govori kako mali pomak oko težine $w_{jk}^{(L)}$ utječe na funkciju gubitka za određeni primjerak treniranja. U stvarnosti imamo cijeli skup od n podataka kojim treniramo našu neuronsku mrežu. Tada iz svih n derivacija funkcija gubitaka C_i (gdje $i = 0, 1, ..., n$) uzimamo aritmetičku sredinu, tj. imamo:

$$\frac{\partial C}{\partial w_{jk}^{(L)}} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial C_i}{w_{jk}^{(L)}} \tag{3.26}$$

Računanje derivacija funkcije gubitaka po težinama koje su u dubljim slojevima mreže će direktno ovisiti o računanju istih derivacija u svakom prethodnom sloju. Počinjemo od zadnjeg sloja i uzastopno primjenjujući pravilo ulančanog deriviranja, idemo prema natrag sve do ulaznog sloja. Ovo je važan dio učenja neuronske mreže.

3.2.8 Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (engl. *Convolutional Neural Networks, CNN*) koriste proces konvolucije u barem jednom od svojih slojeva [20]. Redovito se koriste kod računalnog vida i procesiranja slika. Cilj svakog CNN-a je izvlačenje i učenje korisnih uzoraka na slici koju razmatramo kroz proces konvolucije, dodjeljivanje težina manjim uzorcima te prepoznavanje i razlikovanje većih uzoraka. Arhitektura CNN-a je slična uzorcima veza neurona u vizualnom režnju ljudskog mozga. I kod ovih mreža koristimo neurone koji imaju težine, ulazne informacije, izlazne vrijednosti, funkciju gubitka i propagaciju unatrag.

Neka je slika koju analiziramo u boji, a boje su definirane kao RGB vrijednosti (engl. *Red, Green, Blue*). Ako se radi o slici koja ima 300×300 piksela, onda jedan neuron u prvom skrivenom sloju ima $300 \times 300 \times 3 = 270000$ težinskih parametara. U tom sloju ima više neurona, a mreže često imaju više skrivenih slojeva. Takve neuronske mreže postaju računalno vrlo zahtjevne. Zbog toga koristimo proces konvolucije. Uloga konvolucijskog sloja je da sažme sliku u oblik koji je računalu lakše obraditi, a bez gubljenja bitnih obilježja koja su važna za donošenje odluka.

Konvolucija u matematici je operacija između dvije funkcije f i g koja preslikava

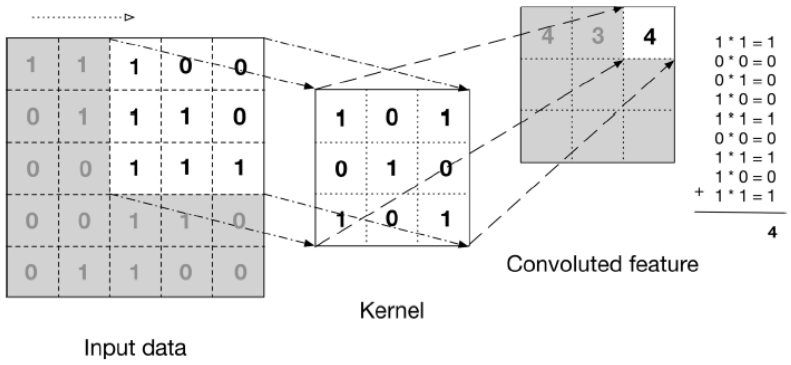
jednu funkciju na drugu i uzima integral umnoška slika tih funkcija u svakoj točki domene. Neka su f i g glatke funkcije na $\mathbb{R}$ i neka ovise o parametru t. Konvoluciju od f i g definiramo kao [31]:

$$f*g = \int_{-\infty}^{+\infty} f(t)g(\tau - t)dt \tag{3.27}$$

Ako imamo domenu funkcija f i g samo na intervalu od $[0, \infty)$ tada se integral svodi na:

$$f*g = \int_{0}^{t} f(t)g(\tau - t)dt \qquad f, g : [0, \infty) \to \mathbb{R} \tag{3.28}$$

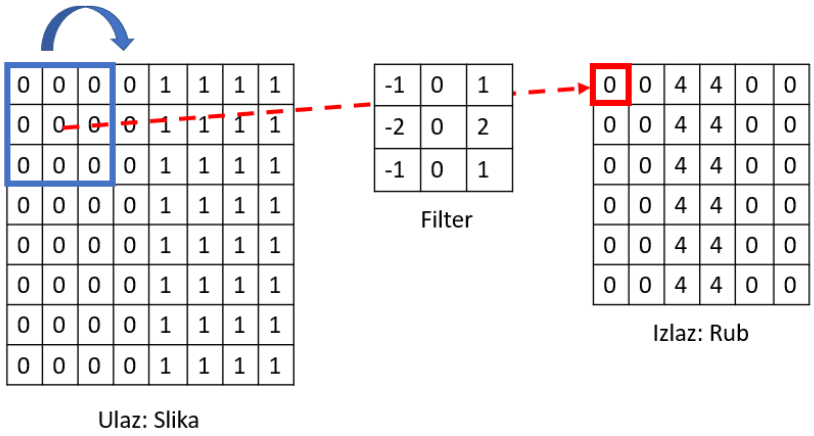
Intuitivno, ovu matematičku operaciju možemo promatrati kao proces u kojem pratimo kontribucije funkcija f i g po nekom parametru t. Konvolucija je kontinuirana verzija množenja. Proces konvolucije se široko koristi u obradi signala i igra ulogu pravila po kojem se informacije sažimaju.



Slika 3.10: Shematski prikaz procesa konvolucije [20]

Proces konvolucije u CNN-u daje kao izlaz sve precizniju mapu obilježja slike. Često konvolucijski sloj nazivamo filterom jer se kroz taj proces dobivaju točno određena obilježja slike, na primjer rubovi. Slika 3.10 prikazuje jedan jednostavan korak u procesu konvolucije. U svakom koraku procesa konvolucije imamo matricu ulaznih podataka i jezgru, tj. filter. Filterom određujemo koja svojstva slike želimo izvući. Na primjer možemo imati filter samo za horizontalne rubove, samo za vertikalne rubove, itd. Na Slici 3.11 vidimo proces konvolucije filterom za vertikalni rub.

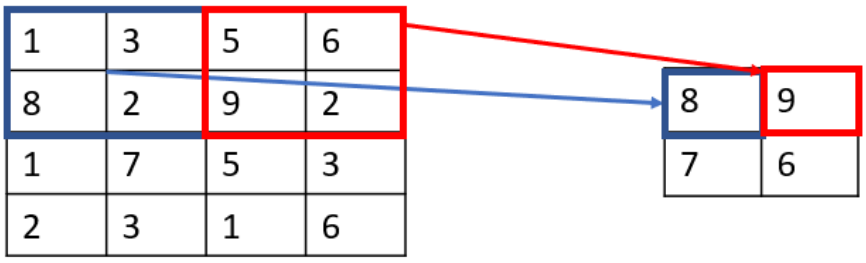
Možemo zamisliti da su 1 svijetli pikseli, a 0 tamni. Tada se rub nalazi na mjestu dva stupca kojeg sačinjavaju četvorke. Te četvorke i nule dobivamo tako što kvadratnu podmatricu dimenzija 3×3 množimo filterom koristeći pravilo klasičnog množenja matrica. Dobivenu vrijednost postavili smo u prvi redak i prvi stupac izlaza. Zatim smo kvadratnu podmatricu pomakli jedno mjesto udesno i ponovili postupak.



Slika 3.11: Shematski prikaz procesa primjene filtera u procesu konvolucije [32]

Sažimanje (engl. *pooling*) je proces koji slijedi nakon konvolucije s ciljem smanjivanja podataka koje mreža treba procesirati. Uobičajeno je da nakon svakog konvolucijskog sloja u mreži imamo jedan sloj sažimanja. Problem izlaznih vrijednosti konvolucijskih slojeva je da su dobivena obilježja osjetljiva. Tako možemo imati obilježja koja su zapravo ista, ali ih naša mreža tretira kao da su različita. Logično je izvući dominantna obilježja koja su invarijantna na translaciju i rotaciju. Sažimanjem sumiramo i sažimamo sva dobivena obilježja te se rješavamo viška. Dvije najčešće metode sažimanja su srednje sažimanje i maksimalno sažimanje (engl. *max pooling*). Prvi od njih uzima srednju vrijednost prisutnosti određenog obilježja, a drugi obilježje koje je najviše aktivirano. U ovom radu koristi se maksimalno sažimanje.

Koraci maksimalnog sažimanja prikazani su na Slici 3.12. Veličina filtera je broj stupaca i redaka filtera kojim ćemo ići po konvoluiranoj izlaznoj matrici (plavi i crveni kvadrat na lijevoj matrici). Veličina koraka je broj stupaca i redaka za koji pomičemo filter nakon svakog koraka. Izabrane veličine filtera i koraka su 2. U svakom koraku biramo najveći mogući iznos u podmatrici i prebacujemo ga u novu matricu. Zatim pomičemo filter za veličinu koraka. Ovim procesom dobivamo maksimalno saženu izlaznu matricu.



Slika 3.12: Shematski prikaz maksimalnog sažimanja

4 Klasifikacija oblika galaksija primjenom neuronskih mreža

4.1 Korišteni alati i metode

Python Python je programski jezik visoke razine apstrakcije koji je moguće koristiti u svim područjima računalne znanosti [33]. Nalazi se pri vrhu liste najpopularnijih programskih jezika. Python je jezik opće namjene, ali ima vrlo jednostavnu sintaksu te ga mogu koristiti i programeri početnici. Brzina programiranja i jednostavna sintaksa čini ga privlačnim ljudima iz raznolikih znanstvenih sfera kojima struka nije programiranje, ali se bave obradom podataka i kompleksnim računalnim analizama. Python je kreirao Guido van Rossum, još davne 1990. godine, a ime dobiva po poznatoj televizijskoj seriji Monty Python's Flying Circus. Python je objektno orijentiran programski jezik. Postoji mnogo dobro testiranih biblioteka koje su na raspolaganju programerima. Neke od tih biblioteka se koriste u ovom radu.

Jupyterova bilježnica Jupyterova bilježnica je interaktivna web aplikacija otvorenog koda koja nam omogućava da dijelimo i razvijamo mrežno-bazirane dokumente koji sadrže programski kod, jednadžbe, vizualizacije podataka i tekst [34]. One se mogu spojiti na više različitih jezgri operacijskog sustava i izvoditi programe u različitim programskim jezicima. Također, sadržaj bilježnice možemo spremati u raznim oblicima, kao što su pdf i html dokumenti. Jupyterove bilježnice su pogodne za razne kompleksne zadatke iz područja podatkovne znanosti, statističko modeliranje, strojno i duboko učenje. Zato su u danas u širokoj upotrebi.

TensorFlow TensorFlow je platforma alata, biblioteka i resursa koja omogućava razvoj aplikacija koje koriste strojno učenje [35]. Stvorio ga je tim Google Brain. Viša razina apstrakcije biblioteke je napisana u programskom jeziku Python i kroz njega pruža prikladno aplikacijsko sučelje za programiranje (API). Sve u pozadini se izvodi u superbrzom i efikasnom C++ programskom jeziku. TensorFlow možemo pokrenuti na bilo kojoj okolini, bio to lokalni stroj, klaster u oblaku (engl. *cloud*), na iOS i Android uređajima, bilo kojem procesoru ili grafičkoj kartici. Google za korisnike Google Cloud-a omogućava pokretanje TensorFlowa na TensorFlow Processing Units

(TPU) za još veću brzinu procesiranja.

Keras Keras je biblioteka s licencom otvorenog koda koja pruža Python sučelje za korištenje neuronskih mreža i platforme TensorFlow [36]. U svojim prethodnim verzijama Keras se koristio i kao sučelje za Microsoft Cognitive Toolkit, Theano i PlaidML biblioteke. Od verzije 2.4 podržan je samo TensorFlow. Keras sadrži brojne implementacije često korištenih sastavnica neuronskih mreža, kao što su slojevi, aktivacijske funkcije i optimizatori. Znatno pojednostavljuje rad s procesiranjem slika i teksta. Uz standardne, Keras sadrži konvolucijske i povratne neuronske mreže. Kod je dostupan na GitHub repozitoriju [37].

Matplotlib Matplotlib je biblioteka za crtanje i vizualizaciju grafova u programskom jeziku Python [38]. Dizajniran je tako da je često nekoliko linija koda dovoljno za realizaciju grafa koji nam je potreban. Pored toga, Matplotlib podržava puno naprednih opcija. Zbog toga je vrlo pogodan za sve znanstvene discipline.

Pandas Pandas je alat za brzo i jednostavno manipuliranje i analizu velike količine podataka [39]. To je biblioteka napisana u Pythonu. Pandas je počeo pisati Wes McKinney u 2008. godini. Danas je to jedna od najpopularnijih biblioteka u Pythonu. Specijalno je korisna za rad s numeričkim tablicama. Pandas je poznata po svojim jednostavnim objektima za spremanje podataka kao što su Series za nizove podataka svih tipova i DataFrames za tablično prikazane podatke. Pandas može čitati podatke različitih formata, npr. JSON, SQL, CSV i Microsoft Excel.

4.2 Skup podataka

U ovom radu opisana je klasifikacija slika galaksija u vidljivom dijelu elektromagnetskog spektra primjenom strojnog učenja. Obilježje koje analiziramo je oblik. Podijelit ćemo galaksije, kao što je opisano u poglavlju 2, na tri klase: spiralne, eliptične i lećaste. Skup podataka koji koristimo nalazi se na jednoj od stranica Googleove zajednice za strojno učenje Kaggle [40]. Motivacija za metode analize i računa koje su opisane u ovom diplomskom radu je blog doktora računalne znanosti Nuno Ramos Carvalha, koji je predložio model [41]. Programski kod u Pythonu koji je korišten

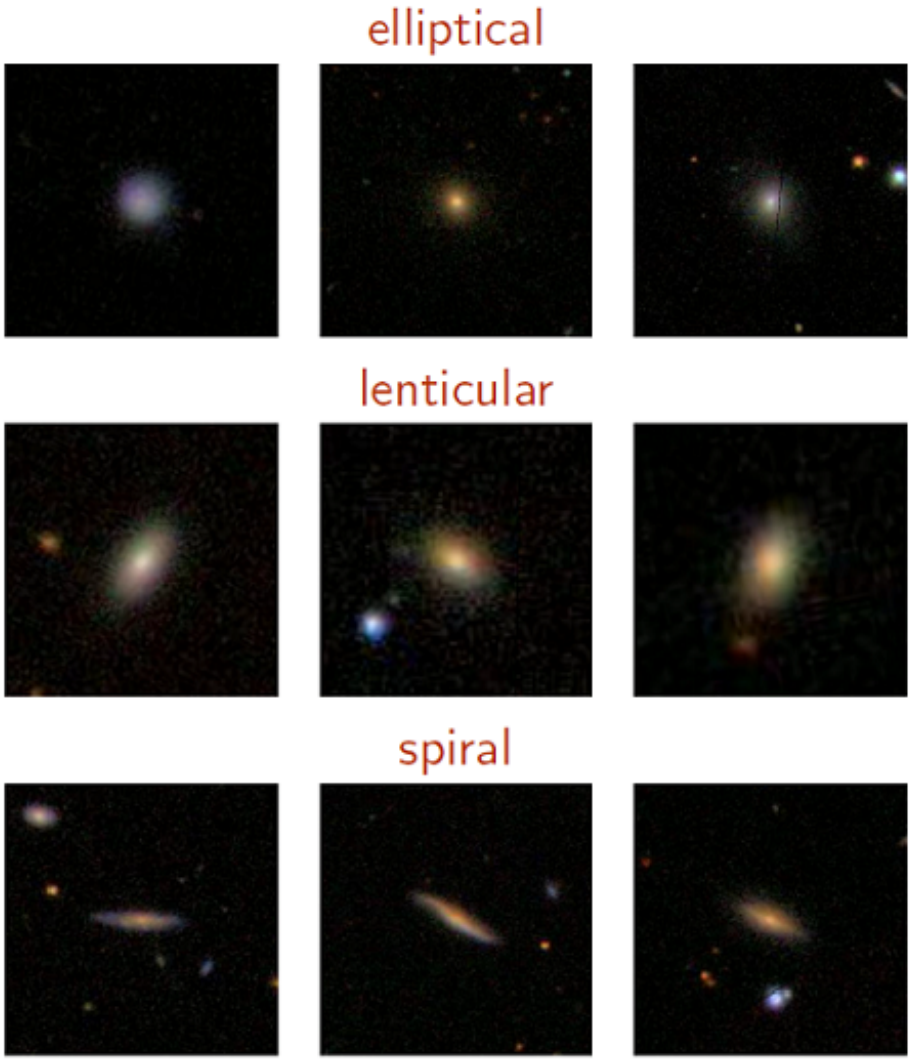
za definiranje i treniranje modela nalazi se u repozitoriju na njegovoj GitHub stranici [42].

Zadatak na portalu Kaggle koristi podatke iz Galaxy Zoo projekta [43]. Galaxy Zoo je jedan od projekata znanosti za građanstvo sa platforme Zoouniverse koja okuplja znanstvenike i amatere [44]. Platforma olakšava svima da sudjeluju u istraživanjima na akademskoj razini i tako ubrzavaju velike znanstvene projekte. Galaxy Zoo je projekt klasifikacije velikog broja novootkrivenih galaksija. Projekt do sada (početak 2022. godine) ima preko 78 000 volontera i preko 3,9 milijuna klasificiranih galaksija. Galaxy Zoo postoji već 14 godina. U prvim godinama rada su analizirane galaksije iz Sloanovog digitalnog pregleda neba (SDSS) [10] i Hubbleovog teleskopa [15]. Najnovije slike galaksija koje analizira GalaxyZoo su rezultat projekta Dark Energy Camera Legacy Survey (DECaLS) [45]. Postoji nekoliko generacija Galaxy Zoo projekta. U ovom radu se koriste slike galaksija iz Galaxy Zoo 2 projekta.

Prikupljanje podataka je jednako važno kao klasifikacija i označavanje tih istih podataka. U svim područjima primjene nadziranog strojnog učenja označavanje podataka uzima puno vremena. Modeli strojnog učenja ovise o podacima koje imamo. Dobra obrada podataka je vrlo važan korak u treniranju svake neuronske mreže. Početni skup podataka u zadatku na portalu Kaggle sadrži 61578 slika. Za svaku sliku volonteri su trebali odgovoriti na pitanja koja su pomagala odrediti karakteristike galaksija, kao što su spiralna obilježja, ili obilježja nalik disku [46]. Pretpostavljeno je da je galaksija eliptična ako je barem 80% sudionika smatralo da je galaksija glatka i barem 40% njih smatralo da je potpuno okrugla. Slični kriteriji su postaviljeni i za druge dvije vrste galaksija. Na Slici 4.1 prikazani su primjeri oblika galaksija.

Uvodimo dva skupa podataka: skup za treniranje i testni skup. Skup podataka za treniranje koristimo kako bi trenirali algoritam. Na taj način pomažemo algoritmu da uči i predviđa rezultate. Ovaj skup podataka uključuje ulazne podatke i očekivane izlazne podatke [47]. U ovom procesu dolazi do prilagodbe pojednih težina između neurona. Skup podataka za treniranje sadrži veći dio ukupnih podataka (oko 60%). Na testnom skupu podataka ispitujemo koliko dobro je naš algoritam istreniran. U testnom skupu ne smiju biti podatci koji su u skupu za treniranje jer bi algoritam znao tražene odgovore.

Početni skup podataka izgradjen je na temelju procjene volontera o morfologiji



Slika 4.1: Primjeri za tri klase galaksija [41]

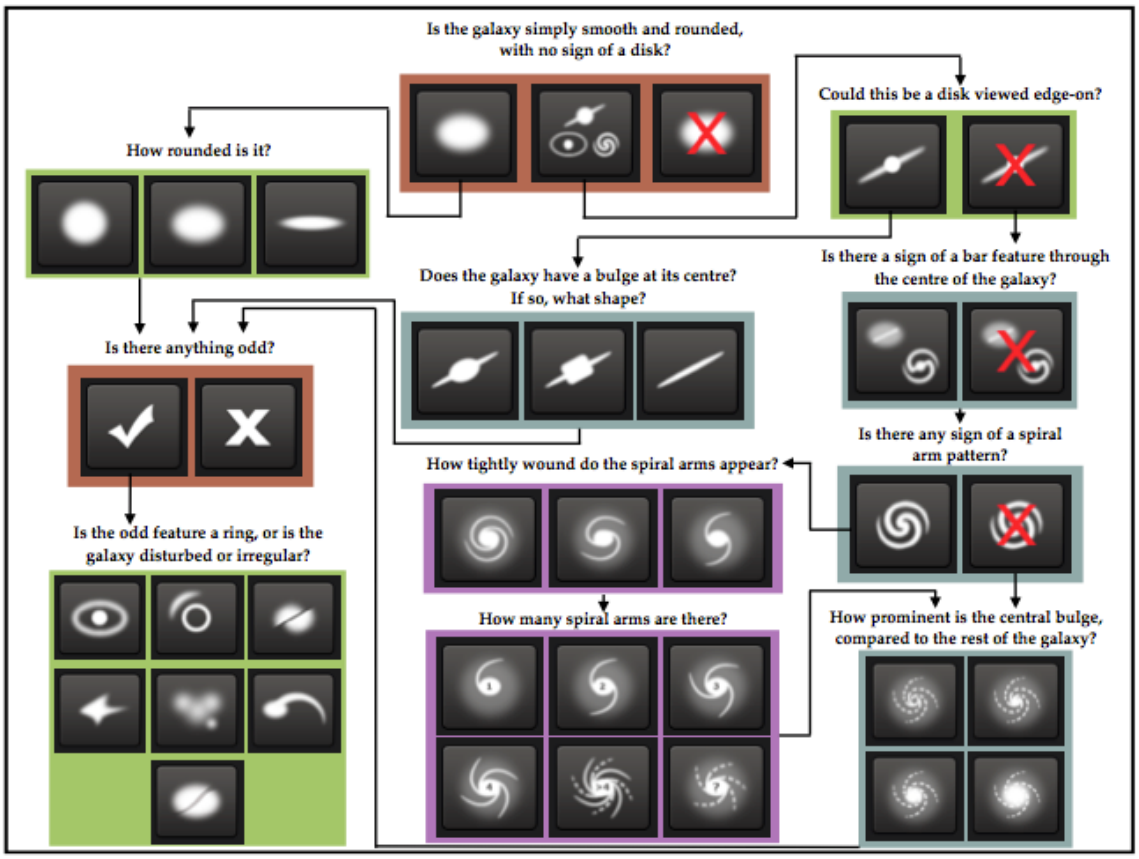
galaksija [46]. Opažanja volontera i njihove procjene o vjerojatnosti pripadnosti galaksije jednom od tri oblika pripremljeni su kao velika CSV tablica. Ta tablica je dobivena iz skupa pitanja na koja su volonteri trebali odgovarati kako bi što bolje opisali svaku od 61578 galaksija. Dio te tablice, nakon obrade, je prikazan na Slici 4.2. Ovakav prikaz tablice omogućila je Pythonova biblioteka pandas i njena metoda $df.head()$ koja daje samo prvih nekoliko stupaca i redaka u tablici. Iz te tablice se moze razviti stablo odlučivanja koje će razvrstati sve slike galaksija u tri tražene skupine: eliptične, spiralne ili lećaste.

Prvi stupac Tablice 4.2 je GalaxyID koji je u ovom zadatku slučajno generirani broj koji omogućava pridruživanje određene vjerojatnosti svakoj slici galaksije. Stupci od Q1.* do Q11.* predstavljaju odgovore volontera. To su brojevi u intervalu $[0, 1]$. Vrijednosti blizu jedinice pokazuju da je veliki broj volontera zaključio da galaksija s

	GalaxyID	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q3.1	Q3.2	Q4.1	Q4.2	...
0	100008	0.383147	0.616853	0.000000	0.000000	0.616853	0.038452	0.578401	0.418398	0.198455	...
1	100023	0.327001	0.663777	0.009222	0.031178	0.632599	0.467370	0.165229	0.591328	0.041271	...
2	100053	0.765717	0.177352	0.056931	0.000000	0.177352	0.000000	0.177352	0.000000	0.177352	...
3	100078	0.693377	0.238564	0.068059	0.000000	0.238564	0.109493	0.129071	0.189098	0.049466	...
4	100090	0.933839	0.000000	0.066161	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...

Slika 4.2: Dio tablice s podacima o galaksijama nakon procjena volontera

izabranim brojem GalaxyiD ima određenu morfološku osobinu. Na temelju odgovora svaka je galaksija svrstana u jednu od kategorija pomoću stabla odlučivanja. Na Slici 4.3 je prikazano stablo odlučivanja projekta Galaxy Zoo 2. Pitanja ima jedanaest i svako pitanje ima različit broj mogućih odgovora. Prvo pitanje ($Q1$) ispituje mišljenja volontera o tome je li objekt na slici gladak, ili ima disk, ili izgleda kao zvijezda. Na to pitanje postoji tri moguća odgovora te zato Tablica 4.2 ima stupce $Q1.1$, $Q1.2$ i $Q1.3$. Drugo pitanje je vidimo li galaksiju s ruba (engl. *edge-on*) ili ne. Odgovora su dva: $Q2.1$ i $Q2.2$. Stupaca Q tipa (tj. odgovora na pitanja) ima 37, kao što je prikazano na Slici 4.3.



Slika 4.3: Prikaz stabla odlučivanja projekta Galaxy Zoo 2 [48]

Za stablo odlučivanja vrijedila su pravila:

$$Eliptične: \quad Q1.1 > 0.8 \quad \& \quad Q7.1 > 0.4 \tag{4.1}$$

$$Lećaste: \quad Q1.1 > 0.8 \quad \& \quad Q7.2 > 0.4 \tag{4.2}$$

$$Spiralne: \quad Q1.2 > 0.8 \quad \& \quad Q2.1 > 0.4 \tag{4.3}$$

Rezultat primjene ovog stabla odlučivanja je:

- Ukupan broj eliptičnih galaksija: 4555
- Ukupan broj lećastih galaksija: 3861
- Ukupan broj spiralnih galaksija: 3078

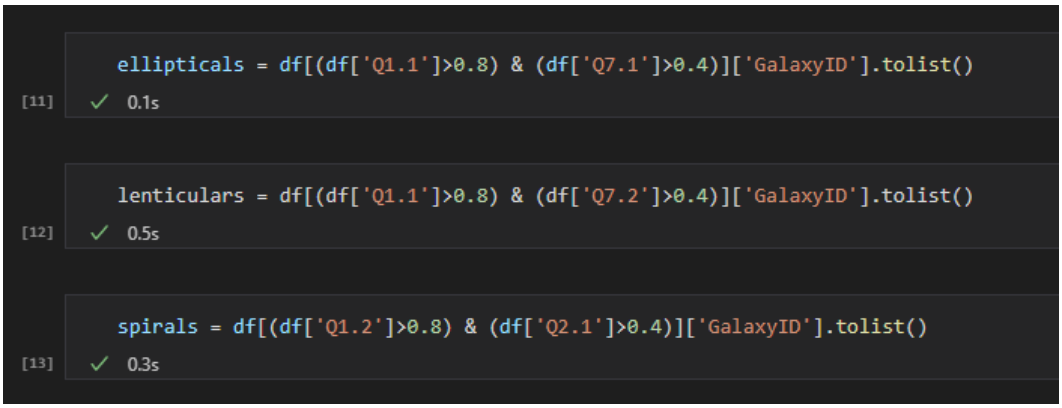
Nakon toga slike galaksija se mogu nasumično podijeliti na skup za treniranje i u testni skup. Svaki je od njih ima po tri direktorija za svaku klasu galaksija. Skup za treniranje sadrži:

- Ukupan broj eliptičnih galaksija: 3188
- Ukupan broj lećastih galaksija: 2702
- Ukupan broj spiralnih galaksija: 2154

Testni skup sadrži:

- Ukupan broj eliptičnih galaksija: 1367
- Ukupan broj lećastih galaksija: 1159
- Ukupan broj spiralnih galaksija: 924

Primjetimo da je u skupu za treniranje više od 60% slika galaksija. Programski kod koji rasporedjuje slike galaksija u pripadnu skupinu je prikazan na Slici 4.4. Implementiran je prema uvjetima 4.1, 4.2 i 4.3.



Slika 4.4: Sortiranje slika galaksija u programskom jeziku Python

4.3 Model neuronske mreže

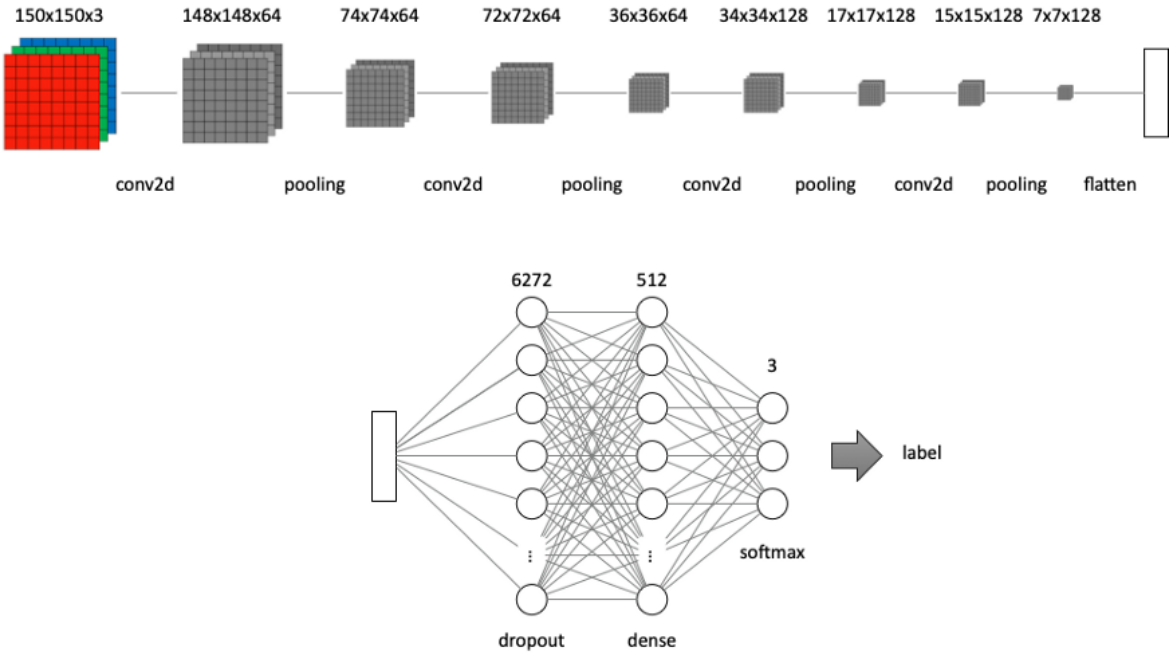
Konstrukcija modela dubokog učenja je zahtjevan korak. Koristi se konvolucijska neuronska mreža (CNN) koja se trenira tako da prepoznaje oblik galaksije na slikama. U ovom radu korištena je verzija 2.3.0 paketa TensorFlow. U pripremi podataka i konstrukciji mreže pomaže i biblioteka Keras koja je, kao i TensorFlow, opisana u poglavlju 4.1.

Model uzima serije (engl. *batch*) podataka (u našem slučaju to su slike) i radi u epohama (engl. *epoch*). Serija je jedan od hiperparametara modela. Hiperparametri algoritma strojnog učenja su varijable koje se definiraju prije primjene tog algoritma na nekom skupu podataka. Ukupan broj slika iz skupa za treniranje se dijeli na manje serije. Na taj način model može obrađivati više slika istovremeno. Što više imamo računalne snage (RAM memorija i memorija jedne ili više grafičkih kartica) to veću seriju slika možemo predati modelu odjednom. Zato je uvijek važno prilagoditi veličinu serije računalnoj snazi koja nam je dostupna. Između procesiranja svake serije događa se podešavanje unutarnjih parametara modela. U ovom radu veličina serije je 32. Epoha je iteracija u kojoj model obradi sve podatke iz skupa. Broj epoha je još jedan od hiperparametara modela koji označava koliko će puta svi podaci proći kroz mrežu. Broj epoha je često veliki (stotine ili čak tisuće). U ovom radu koristimo 100 epoha. Povećanje broja epoha smanjuje greške u predviđanjima modela.

Prvi korak prije definiranja modela je povećavanje skupa podataka (engl. *data augmentation*). U podatkovnoj znanosti, povećanje skupa podataka odnosi se na dodavanje novih podataka skupu i to malim promjenama na već postojećim podacima. Neki od primjera za slike su: rotacija, zrcalna refleksija, modifikacija boja, zumiranje i izrezivanje pojedinih dijelova. Ovo je jedan od važnih alata kojim se nastoji spriječiti

prenaučenost modela. Prenaučenost (engl. *overfitting*) je pojava u kojoj model daje gotovo savršene rezultate, ali za točno određenu vrstu podataka. Recimo da želimo naučiti algoritam da prepoznaje psa na slici. Ako uzmemo slike pasa iste pasmine, koji su svi relativno slični, model će vrlo dobro moći prepoznati tu vrstu psa. Ali kada mu damo sliku psa druge pasmine, neće je moći prepoznati jer je treniran na uskom skupu podataka. To nastojimo izbjeći. U ovom radu se od metoda augmentacije podataka koristiti reskaliranje, rotacija, povećanje širine i duljine, horizontalno preokretanje i zumiranje za određeni interval u odnosu na originalnu sliku.

Shematski je model prikazan na na Slici 4.5 gdje se vidi uzastopni sljed konvolucijskih slojeva i slojeva sažimanja. Počinje se sa slikom dimenzija 150×150 piksela.



Slika 4.5: Shematski prikaz modela [41]

Svaki piksel daje kombinaciju od tri boje po RGB (engl. *Red, Green, Blue*) modelu, gdje imamo crvenu, zelenu i plavu kao glavne boje. One u kombinaciji daju cijelu paletu drugih boja. Sliku predajemo prvom konvolucijskom sloju. Nakon svakog konvolucijskog sloja dolazi sloj maksimalnog sažimanja. Maksimalno sažimanje smanjuje rezoluciju izlazne matrice i samim time računalno opterećenje mreže. Također, ovakav oblik sažimanja smanjuje prenaučenost. Uzimamo najaktiviranije piksele i cijela slika prolazi kroz filter za najveće vrijednosti. Tako se rješavamo slabo aktivnih i neaktivnih piksela. Nakon posljednjeg sloja sažimanja dolazi sloj ravnanja (engl. *flatten*) u kojem se podaci iz višedimenzionalnog polja pretvaraju u jednodimenzi-

onalno. To radimo kako bismo novi 1-D vektor mogli predati izlaznom sloju neuronske mreže. U sloju izbacivanja (engl. *dropout*) neuronske mreže nasumično se zanemaruju neki neuroni tokom treniranja modela. Ovo radimo kako bismo poboljšali generalizaciju tokom učenja mreže. Kada ugasimo jedan neuron, drugi mora donositi zaključke umjesto tog neurona. Vjeruje se da tehnika izbacivanja doprinosi razvoju više različitih reprezentacija unutar neuronske mreže, što spriječava prenaučenost modela [49].

U konvolucijskom sloju se kao aktivacijska funkcija koristi zglobnica, ili ispravljena linearna jedinica (engl. *rectified linear unit, ReLU*) koja je prikazana na Slici 3.5. U zadnjem sloju neuronske mreže koristi se softmax aktivacijska funkciju prikazana na Slici 3.6. Ovu funkciju koristimo na izlazu neuronske mreže kako bismo odredili kolika je vjerojatnost da slika pripada jednoj od tri klase galaksija.

Treniranje počinje nakon definiranja modela i pripreme podataka. Prethodno je definirana funkcija gubitka, optimizator i metrika pomoću metode $compile()$ iz klase $tf.keras.model$. Za funkciju gubitka izabrana je kategorična unakrsna entropija. Takva funkcija gubitka koristi se (kao i softmax aktivacijska funkcija) kod klasifikacije podataka u N različitih klasa. Ova funkcija gubitka mjeri koliko je točan izlaz modela čije su izlazne vrijednosti između 0 i 1. Gubitak je veći kad je predviđena vrijednost dalje od očekivane. Gubitak se računa prema formuli [50]:

$$C = -\sum_{i=1}^{N} y_i \cdot log(y_i) \tag{4.4}$$

gdje je C gubitak, a y_i je i-ta skalarna vrijednost u izlaznom vektoru neuronske mreže. Za naš slučaj od tri kategorije u koje možemo svrstati galaksije, funkcija gubitka je:

$$C = -\sum_{i=1}^{3} y_i \cdot log(y_i) = -y_1 log(f(s_1)) - y_2 log(f(s_2)) - y_3 log(f(s_3)) \tag{4.5}$$

gdje je f softmax aktivacijska funkcija, a s_i skalarne vrijednosti vektora koji ulazi u softmax funkciju. Generalno, unakrsna entropija koristi se kako bi se odredila razlika između raspodjela vjerojatnosti i procijenilo koliko su predviđanja vjerojatnosti daleko od očekivanih.

Koristi se optimizacijski algoritam Adam koji je generalizacija stohastičkog gradijentnog spusta [51]. Taj algoritam se koristi umjesto klasičnog stohastičkog gradi-

jentnog spusta za iterativno podešavanje težinskih parametara. Kod klasičnog gradijentnog spusta brzina učenja je konstantna tokom treniranja modela. Adam je kombinacija dva algoritma optimizacije - RMSprop i stohastičkog gradijentnog spusta sa zaletom. Adam (engl. *adaptive moment estimation*) ima prilagodljivu brzinu učenja, što znači da računa pojedinačne brzine učenja u ovisnosti o određenim parametrima. Adam koristi procjene prvog i drugog momenta gradijenta kako bi prilagođavao brzine učenja. N-ti moment slučajne varijable računa se kao očekivanje te varijable na n-tu potenciju ili formalno:

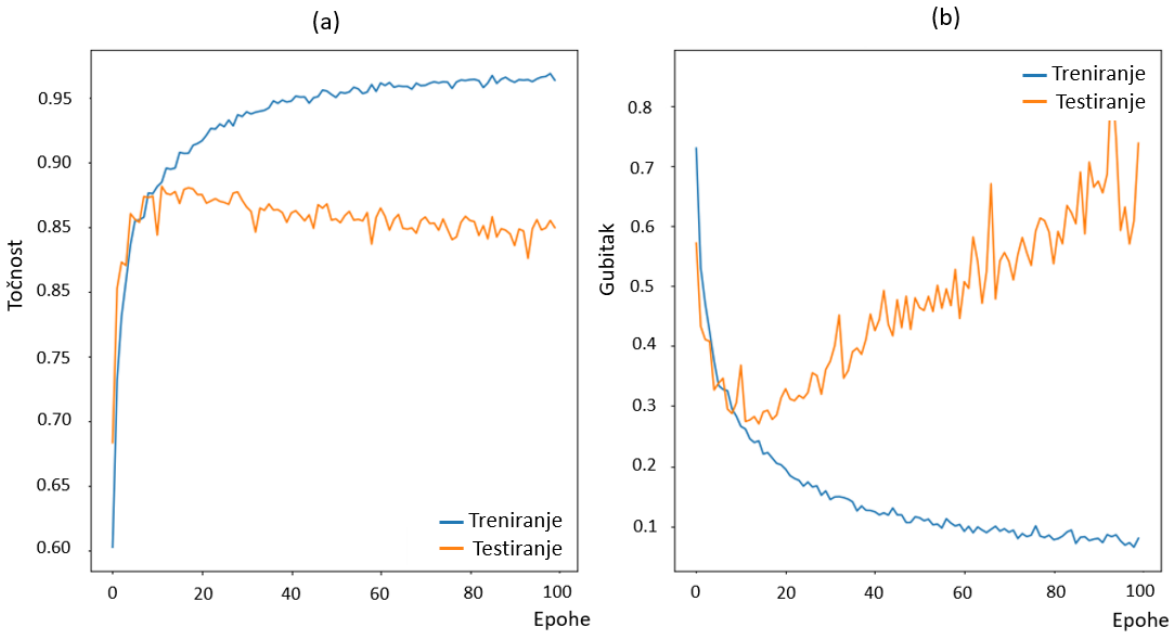
$$m_n = E[X^n] \tag{4.6}$$

Može se primjetiti da je gradijent funkcije gubitka slučajna varijabla. Njegov prvi moment je očekivanje, a drugi moment je varijanca. Metrika je postavljena na točnost, što znači da algoritam prati koliko puta su predviđanja jednaka pravim oznakama na slikama galaksija.

4.4 Rezultati i analiza

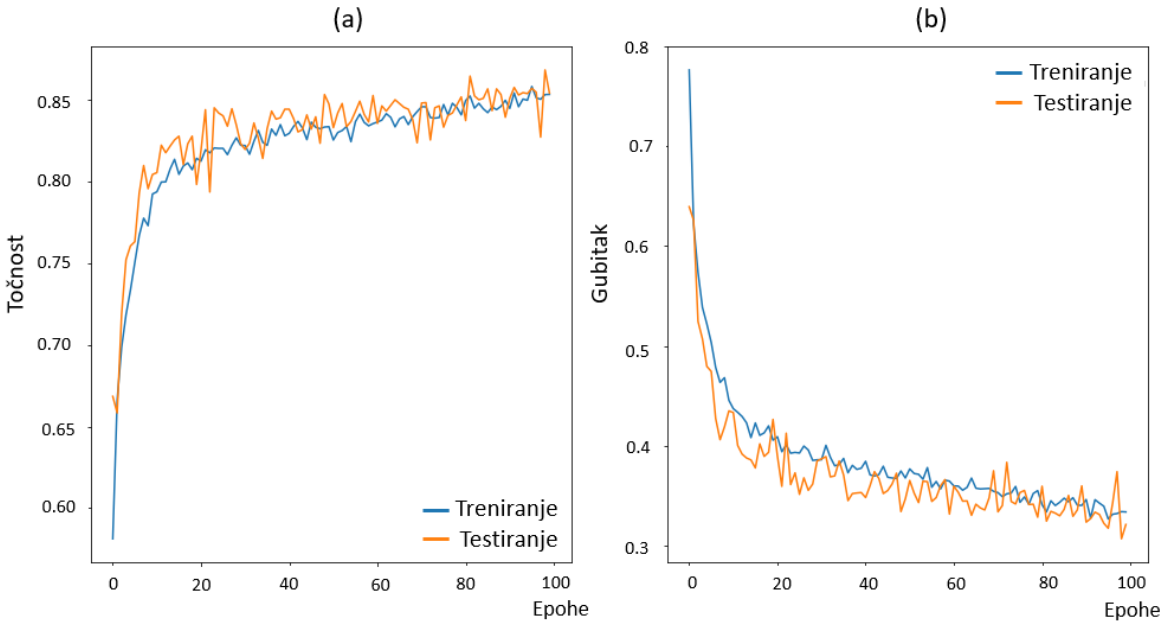
Model se trenira prije i nakon povećanja skupa podataka. Slika 4.6 prikazuje točnost (engl. *accuracy*) i funkciju gubitka na skupu podataka za treniranje i testiranje nakon 100 epoha za početni skup podataka. Točnost definiramo kao omjer broja pogođenih predviđanja i ukupnog broja predviđanja. Funkcija gubitka je definirana kao kategorična unakrsna entropija (jednadžba 4.4). Sa Slike 4.6(a) može se uočiti da postoji prenaučenost. Model postaje jako dobar u predviđanju slika iz skupa za treniranje, ali na testnom skupu točnost opada. To se događa jer se model specijalizira za trenirajući skup, a ne uči tako da može generalizirati naučeno na bilo koju sliku galaksije.

Bolji rezultati se mogu dobiti na većem skupu podataka. Skup podataka je povećan nasumičnim transformacijama na slikama kao što su rotacija, zumiranje i zrcaljenje. Na ovaj način povećava se varijabilnost unutar skupa podataka za treniranje. Omogućavamo modelu da izvuče više korisnih informacija kako bi mogao davati bolja predviđanja na raznovrsnijem skupu podataka. Grafovi nakon povećanja skupa za treniranje prikazani su na Slici 4.7. Možemo uočiti da za trenirajući skup podataka, kao i za testni skup, točnost raste s porastom broja epoha. Nema stagnacije i pada na testnom skupu, kao što je to slučaj za manji skup podataka.



Slika 4.6: Točnost (a) i funkcija gubitka (b) na skupu podataka za treniranje i testiranje prije povećavanja skupa podataka

Model je provjeren s nekoliko slika galaksija koje nisu bile prisutne niti u testnom, niti u skupu podataka za treniranje. Neki od rezultata prikazani su na Slikama 4.8 i 4.9. Oznaka $[1, 0, 0]$ u programu predstavlja eliptičnu, a $[0, 0, 1]$ spiralnu galaksiju.



Slika 4.7: Točnost (a) i funkcija gubitka (b) na skupu podataka za treniranje i testiranje nakon povećavanja skupa podataka

Slika 4.8: Provjera modela: [1,0,0] - eliptična galaksija [41]

Slika 4.9: Provjera modela: [0,0,1] - spiralna galaksija [41]

5 Zaključak

U ovom radu analizirana je klasifikacija oblika galaksija primjenom konvolucijskih neuronskih mreža. Model je treniran na velikom broju slika galaksija koje su vizualno klasificirali brojni znanstvenici i amateri u projektu Galaxy Zoo 2. Zadatak algoritma strojnog učenja je određivanje vjerojatnosti da galaksija pripada eliptičnoj, spiralnoj, ili lećastoj klasi. Početni skup podataka za treniranje dao je relativno dobre rezultate, ali na testnom skupu je postojala prenaučenost. Nakon povećavanja skupa podataka, model je dao vrlo dobre rezultate i na testnom skupu.

Suvremeni teleskopi generiraju ogromne i sve veće količine podataka među kojima se skrivaju važne znanstvene informacije. Za uspješnije otkrivanje tih informacija potrebno je razvijati nove algoritme strojnog učenja, jake procesore i grafičke kartice te memorijske elemente za pohranu. Metode opisane u ovom radu na primjeru klasifikacije slika galaksija mogu se uspješno primijeniti i dalje usavršavati i u drugim područjima analize slika, kao što su npr. rezultati medicinskih istraživanja tkiva i tumora, ili raspoznavanje ljudskih lica.

6 Metodički dio

Nastavna priprema: algoritmi za sortiranje

ŠKOLA: Srednja škola, gimnazija

RAZRED: 2. razred

NASTAVNA JEDINICA: Algoritmi za sortiranje u Pythonu

NAZIV METODIČKE JEDINICE: Računalno razmišljanje i programiranje

PREDVIĐEN BROJ SATI: 2

PREDMETNI ISHODI:

B.2.1 analizira osnovne algoritme s jednostavnim tipovima podataka i osnovnim programskim strukturama i primjenjuje ih pri rješavanju novih problema

B.2.2 u zadanome problemu uočava manje cjeline, rješava ih te ih potom integrira u jedinstveno rješenje problema

B.2.3 rješava problem primjenjujući jednodimenzionalne strukture podataka

B.2.4 analizira sortiranje podataka kao važan koncept za rješavanje različitih problema

RAZRADA PREDMETNIH ISHODA

- Opisuje i primjenjuje standardne algoritme sortiranja i pretraživanja podataka.
- Primjenjuje sortiranje kao dio strategije za rješavanje problema.
- Povezuje različite algoritme s različitom vremenskom složenosti.
- Povezuje vrijeme potrebno za izvođenje programa s veličinom ulaznih podataka.
- Razlikuje brže i sporije algoritme za sortiranje.

MEĐUPREDMETNI ISHODI:

MPT Uporaba IKT

D 4. 1.

- Učenik samostalno ili u suradnji s drugima stvara nove sadržaje i ideje ili preoblikuje postojeća digitalna rješenja primjenjujući različite načine za poticanje kreativnosti.

IKT C.5.1.

- Učenik samostalno provodi složeno istraživanje s pomoću IKT-a.

IKT D.5.2.

- Učenik samostalno predlaže moguća i primjenjiva rješenja složenih problema s pomoću IKT-a.

SUODNOS: Matematika

- Primjenjivanje pravila pri rješavanju različitih zadataka (sa zagradama, pravila rješavanja zadataka riječima), stvaranje nizova i objašnjavanje pravilnosti nizanja.
- Primjena svojstva komutativnosti i veze množenja i dijeljenja, primjena stečenih matematičkih spoznaja o brojevima i svojstvima računskih radnji pri rješavanju problemskih situacija.

VRSTA NASTAVE: Istraživački usmjerena nastava
NASTAVNE METODE:

- Metoda razgovora – usmjerena rasprava,
- Metoda pisanja/crtanja

TIP SATA:

- Vježbe, interaktivno izlaganje

OBLICI RADA:

- Frontalni

NASTAVNA SREDSTVA I POMAGALA:

- Računalo, projektor, Python IDLE, ploča

ZA UČENIKE KOJI ŽELE ZNATI VIŠE:

- Python Tutorial (1999.-2019.), W3Schools, dostupno na: https://www.w3schools.com/python/
- Sorting Algorithms, GeeksforGeeks, dostupno na: https://www.geeksforgeeks.org/sorting-algorithms/

- LeetCode, dostupno na:
  https://leetcode.com

LITERATURA (za nastavnika i učenike):

- Budin, L.; Brođanac, P.; Markučić, Z.; Perić, S. Napredno rješavanje problema programiranjem u Pythonu, udžbenik za prirodoslovno-matematičke gimnazije, Zagreb: Element, 2012.
- Šafar Đerki, D.; Leventić, A.; Ivanović Ižaković, D.; Stjepanek, N.; Tomić, V. SVIJET INFORMATIKE 3 - udžbenik informatike s dodatnim digitalnim sadržajima u trećem razredu gimnazija, Zagreb: Školska knjiga, 2020.
- Dmitrović, N.; Grabusin, S.; Bujanović, Z.; Miletić, Lj.; Kager, D. Informatika 3: Udžbenik informatike za 3. razred prirodoslovno-matematičkih gimnazija. Zagreb : SysPrint, `https://e.udzbenik.hr/pov20/g3/`, 23. 12. 2021.
- Fehiliy, C. Python, Berkeley: Peachpit Press, 2002.
- Zelle, J. Python Programming, Wilsonville: Franklin, Beedle & Associates. Inc., 2004.

TIJEK NASTAVNOG SATA

Uvodni dio sata

Uvodni problem: Zadana je lista u kojoj se nalaze visine 944 najviših zgrada na svijetu izraženo u metrima. Potrebno je ispisati najvišu zgradu.

- **Kako biste pronašli najvišu zgradu?**
- **Kako uspoređujemo visine zgrada?**
- **Koliko puta se uspoređuju podaci?**
- **Što bismo mogli napraviti kako bismo smanjili broj uspoređivanja i prolazaka kroz podatke?**
- **Možemo li nekako izbjeći prolazak kroz cijelu listu?**
- **Ima li ikakvog utjecaja količina podataka?**
- **Što ako trebamo pronaći najvišu zgradu 50 puta dnevno?**

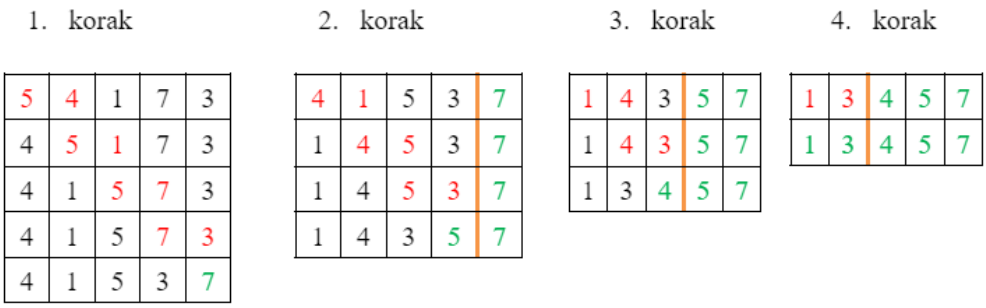
Rješavamo zadatak na klasičan način traženjem maksimuma u listi. Programski kod je prikazan na Slici 6.1. S učenicima raspravljam o **broju koraka** koji je potrebno napraviti da bi se pronašao maksimum. Učenici zaključuju da nije zgodno svaki puta prolaziti kroz cijelu listu kako bi se našla najviša zgrada. Podaci bi se mogli **sortirati**.

```
zgrade = [265.0, 224.5, 272.0, 275.0]

maxi = zgrade[0]
for i in range(len(zgrade)):
    if zgrade[i]> maxi:
        maxi = zgrade[i]

print(maxi)
```

Slika 6.1: Kod sortiranja visina zgrada

Središnji dio sata

Uvesti *sortiranje metodom mjehurića* (engl. *bubble sort*). Opisujem metodu i crtam korake, a nakon prva dva retka sortiram niz zajedno s učenicima (Slika 6.2).



Slika 6.2: Shematski prikaz sortiranja metodom mjehurića

Učenicima prikazujem Youtube video na kojem je metoda mjehurića prikazana kroz tradicionalni ples: `https://tinyurl.com/4wdxz97u`

Učenike pitati za opažanja te raspraviti pojedine korake metode nakon čega se formira njen potpuni opis.

Zadatak 1

Napisati program sortiranja metodom mjehurića zajedno s učenicima (Slika 6.3).

```
def bubbleSort(arr):
    x = arr[:]
    n = len(x)
    k = 1
    zamjena = True
    while zamjena:
        zamjena = False
        for i in range(0, n-k):
            if x[i]>x[i+1]:
                zamjena = True
                x[i], x[i+1] = x[i+1], x[i]
        k+=1
    return x
```

Slika 6.3: Programski kod sortiranja metodom mjehurića

Uvodim novi algoritam: *sortiranje izborom najmanjeg elementa* (engl. *selection sort*) (Slika 6.4). Opisujem metodu i crtam korake, a nakon prva dva koraka sortiram

1. korak	5	4	1	7	3	6	2
2. korak	1	4	5	7	3	6	2
3. korak	1	2	5	7	3	6	4
4. korak	1	2	3	7	5	6	4
5. korak	1	2	3	4	5	6	7
6. korak	1	2	3	4	5	6	7
7. korak	1	2	3	4	5	6	7

Slika 6.4: Shematski prikaz sortiranja izborom najmanjeg elementa

niz zajedno s učenicima. Prozivam učenike tako da svaki korak napravi jedan od učenika.

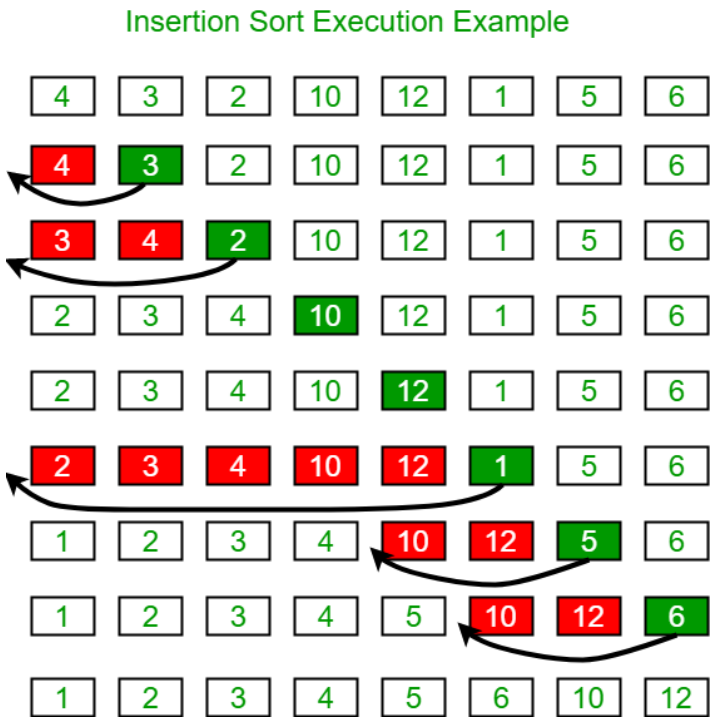
Zadatak 2

Napisati program sortiranja izborom najmanjeg elementa zajedno s učenicima (Slika 6.5).

```
def selectionSort(arr):
    x = arr[:]
    n = len(x)
    for i in range(n):
        mini = i
        for j in range(i+1,n):
            if x[j]<x[mini]:
                mini = j
        x[i],x[mini] = x[mini],x[i]
    return x

x = [5,4,1,7,3,6,2]
print(x)
print(selectionSort(x))
```

Slika 6.5: Programski kod sortiranja izborom najmanjeg elementa

Uvodim *algoritam sortiranja umetanjem* (engl. *insertion sort*) koji je prikazan na Slici 6.6. Zajedno s učenicima prolazim kroz korake sortiranja, prva dva koraka



Slika 6.6: Shematski prikaz sortiranja umetanjem [52]

objašnjavam ja, a nakon tog svaki od koraka izvodi i objašnjava učenik.

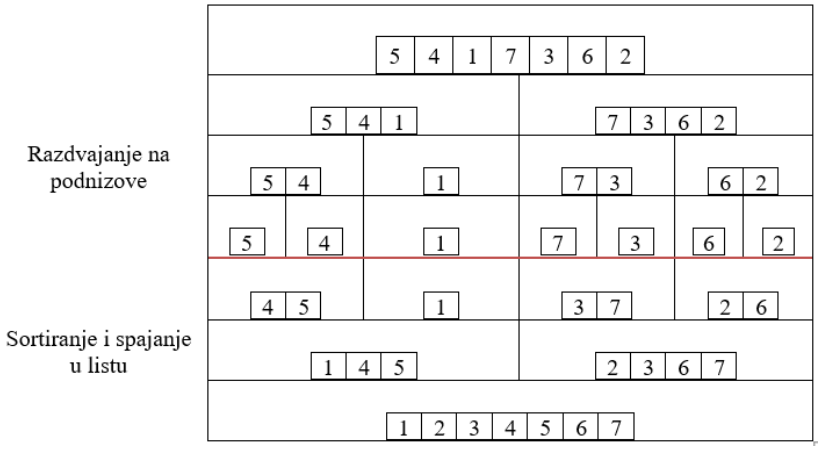
Zadatak 3

Napisati program sortiranja umetanjem zajedno s učenicima (Slika 6.7).

```
def insertionSort(arr):
    x = arr[:]
    n = len(x)

    for i in range(1,n):
        key = x[i]
        j = i-1
        while j>=0 and key<x[j]:
            x[j+1],x[j] = x[j],x[j+1]
            j-=1
    return x


x = [5,4,1,7,3,6,2]
print(insertionSort(x))
```

Slika 6.7: Programski kod sortiranja umetanjem

Uvodim *algoritam sortiranja sjedinjavanjem* (engl. *merge sort*) (Slika 6.8).



Slika 6.8: Shematski prikaz sortiranja sjedinjavanjem

Crtam na ploču, zajedno s učenicima prolazim korake na jednostavnom primjeru.

Zadatak 4

Napisati program sortiranja sjedinjavanjem zajedno s učenicima. Kod je prikazan na Slici 6.9.

Završni dio sata Uspoređujem sortiranje sjedinjavanjem s jednostavnijim algoritmima sortiranja na velikoj listi visina zgrada iz uvodnog problema. Cilj je pokazati učenicima da je vrijeme za slučaj algoritma sortiranja sjedinjavanjem puno kraće nego za metodu mjehurića i metodu izbora najmanjeg elementa. Programski kod usporedbe vremena izvršavanja algoritama sortiranja je prikazan na Slici 6.10.

```
def mergeSort(x,l=0,d=None):
    if d==None:
        d = len(x)
    n = l+d
    sredina = n//2

    if l+1>=d:
        return

    #lijeva podlista
    mergeSort(x, l, sredina)
    #desna podlista
    mergeSort(x, sredina, d)

    i = l #lijevi brojač
    j = sredina #desni brojač
    lista = []

    #prolazi kroz lijevu i desnu listu
    #istovremeno i dodaj manji element u novu
    while(i < sredina and j < d):
        if x[i]<x[j]:
            lista.append(x[i])
            i+=1
        else:
            lista.append(x[j])
            j+=1
    #dodaj u listu što je ostalo
    while(i < sredina):
        lista.append(x[i])
        i+=1
    while(j < d):
        lista.append(x[j])
        j+=1
    #elementi x-a od l do d su sada
    #maloprije sortirana podlista
    x[l:d] = lista

x = [5,4,1,7,3,6,2]
print(x)
mergeSort(x)
print(x)
```

```
def merge(a, b):
    i = j = 0
    s = []
    while i < len(a) and j < len(b):
        if a[i] < b[j]:
            s.append(a[i])
            i+=1
        else:
            s.append(b[j])
            j+=1
    s += a[i:]
    s += b[j:]
    return s

def mergeSort(x):
    if len(x) > 1:
        s = len(x) // 2
        a = mergeSort(x[:s])
        b = mergeSort(x[s:])
        x = merge(a, b)
    return x
```

Slika 6.9: Programski kod sortiranja sjedinjavanjem

```
t1 = time.time()
bubbleSort(zgrade)
t2 = time.time()
dt = t2-t1
print("Bubble sort:", round(dt*1e3,2),"ms")

t1 = time.time()
selectionSort(zgrade)
t2 = time.time()
dt = t2-t1
print("Selection sort:", round(dt*1e3,2),"ms")

t1 = time.time()
mergeSort(zgrade)
t2 = time.time()
dt = t2-t1
print("Merge sort:", round(dt*1e3,2),"ms")
```

Slika 6.10: Kod za mjerenje vremena sortiranja pojedinih algoritama

Literatura

[1] NASA, About - Story: Edwin Hubble, https://tinyurl.com/5a2byt77, 20. 8. 2021.

[2] NASA, James Webb Space Telescope: Galaxies Over Time, https://tinyurl.com/45u6aftc, 20. 8. 2021.

[3] Sparke, L. S.; Gallagher, J. S. III. Galaxies in the Universe: An Introduction. Cambridge: Cambridge University Press, 2000.

[4] Wikipedia, Galaxy, https://en.wikipedia.org/wiki/Galaxy, 12. 9. 2021.

[5] Holwerda, B.W. Galaxy Morphology. Bristol: IOP Publishing, 2022.

[6] Wikipedia, Hubble sequence, https://tinyurl.com/2p8mk24b, 10. 1. 2022.

[7] Wikipedia, IC 1101, https://en.wikipedia.org/wiki/IC_1101, 11. 1. 2022.

[8] Wikipedia, NGC 5866, https://tinyurl.com/4dpvwyu3, 11. 1. 2022.

[9] Wikipedia, Andromeda Galaxy, https://tinyurl.com/2p9f4rv5, 11. 1. 2022.

[10] The Sloan Digital Sky Survey (SDSS), https://www.sdss.org, 21. 10. 2021.

[11] L. Ceraj, Astroučionica: Teleskopi kojima opažamo daleke galaksije, https://tinyurl.com/mryhkwp3, 12. 1. 2022.

[12] Wikipedia, List of radio telescopes, https://tinyurl.com/35pysvts, 12. 1. 2022.

[13] Wikipedia, Infrared telescope, https://tinyurl.com/5cyrrazu, 13. 1. 2022.

[14] Very Large Telescope, https://tinyurl.com/3f865cx8, 13. 1. 2022.

[15] Hubble Space Telescope, https://tinyurl.com/2s3s86t8, 13. 1. 2022.

[16] Chandra X-ray Observatory, https://tinyurl.com/2p83hwks, 13. 1. 2022.

[17] Gamma-ray astronomy, https://tinyurl.com/5n8fw6ur, 13. 1. 2022.

[18] Mitchell, T. M. Machine Learning. New York: McGraw-Hill, 1997.

[19] Géron, A. Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligent systems. 2nd ed. Sebastopol: O'Reilly Media, 2019.

[20] Patterson, J; Gibson, A. Deep Learning, A Practitioner's Approach. Sebastopol: O'Reilly Media, 2017.

[21] Muller, A. C.; Guido, S. Introduction to Machine Learning with Python. Sebastopol: O'Reilly Media, 2017.

[22] Wikipedia, Neuron, https://en.wikipedia.org/wiki/Neuron, 2. 9. 2021.

[23] Lefkowitz, M. Professor's perceptron paved the way for AI – 60 years too soon, https://tinyurl.com/y4qchvbv, 4. 9. 2021.

[24] Wikipedia, Sigmoid function, https://tinyurl.com/msmuud9f, 20. 9. 2021.

[25] J. Brownlee, Softmax Activation Function with Python https://tinyurl.com/ycks8f3j, 21. 9. 2021.

[26] Rumelhart, D.; Hinton, G.; Williams, R. Learning representations by back-propagating errors, Nature, 323, 533–536 (1986).

[27] Wikipedia, Gradient descent, https://tinyurl.com/4wyumant, 8. 10. 2021.

[28] Nielsen, M. A, Neural Networks and Deep Learning. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/, 23.9. 2021.

[29] Deeplizard, Deep Learning Fundamentals: Backpropagation Explained, Mathematical Observations, https://tinyurl.com/2dd9wjxp, 18. 12. 2021.

[30] Deeplizard, Deep Learning Fundamentals: Backpropagation Explained, Calculating The Gradient, https://tinyurl.com/473huw8t, 19. 12. 2021.

[31] Wikipedia, Convolution, https://en.wikipedia.org/wiki/Convolution, 21. 10. 2021.

[32] Isikdogan, L. Convolutional Neural Networks Explained, https://tinyurl.com/3ekdhkex, 17. 11. 2022.

[33] Wikipedija, Python (programski jezik), https://tinyurl.com/mva6sdnn, 12. 10. 2021.

[34] Project Jupyter, https://jupyter.org/, 12. 10. 2021.

[35] TensorFlow, https://www.tensorflow.org, 12. 10. 2021.

[36] Wikipedia, Keras, https://en.wikipedia.org/wiki/Keras, 13. 10. 2021.

[37] Keras GitHub, https://github.com/keras-team/keras, 13. 10. 2021.

[38] Wikipedia, Matplotlib, https://tinyurl.com/yc3d2kjh, 13. 10. 2021.

[39] pandas, https://pandas.pydata.org, 14. 10. 2021.

[40] Kaggle Galaxy Zoo - The Galaxy Challenge, https://tinyurl.com/3krp7w5e, 20. 10. 2021.

[41] N. R. Carvalho, Galaxies Morphology Classification Using Convolutional Neural Networks, https://tinyurl.com/mzsdx8ce, 30. 11. 2021.

[42] N. R. Carvalho, Galaxy ConvNet GitHub, https://tinyurl.com/2p8rafea, 30. 10. 2021.

[43] Galaxy Zoo, https://tinyurl.com/2k2vu9rr, 19. 1. 2022.

[44] Zooniverse, https://www.zooniverse.org, 21. 10. 2021.

[45] The Dark Energy Camera Legacy Survey (DECaLS), https://tinyurl.com/2p8fy63z, 21. 10. 2021.

[46] Galaxy Zoo Classify, https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/classify, 22. 12. 2021.

[47] A. Gonfalonieri, How to Build A Data Set For Your Machine Learning Project, https://tinyurl.com/2p95d4ze, 12. 11. 2021.

[48] Willett, K. W. et al. Galaxy Zoo 2: detailed morphological classifications for 304 122 galaxies from the Sloan Digital Sky Survey, Monthly Notices of the Royal Astronomical Society, 435, 2835-2860 (2013)

[49] J. Brownlee, Dropout Regularization in Deep Learning Models With Keras, https://tinyurl.com/276xphht, 8.2.2022.

[50] R. Gomez, Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names, https://gombru.github.io/2018/05/23/cross_entropy_loss/, 1. 12. 2021.

[51] V. Bushaev, Adam — latest trends in deep learning optimization, https://tinyurl.com/y8dj7cxd, 2. 12. 2021.

[52] GeeksforGeeks, Insertion Sort, https://tinyurl.com/2p8s5xen, 20. 12. 2021.