

Računalni model strojnog učenja za sortiranje otpada

Baronica, Dora

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:235:528266>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2024-07-12**

Repository / Repozitorij:

[Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Dora Baronica

Zagreb, 2023. godina

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

Računalni model strojnog učenja za sortiranje otpada

Mentor:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Dora Baronica

Zagreb, 2023. godina



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE



Središnje povjerenstvo za završne i diplomске ispite
Povjerenstvo za završne i diplomске ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo
materijala i mehatronika i robotika

Sveučilište u Zagrebu	
Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: 602 - 04 / 23 - 6 / 1	
Ur.broj: 15 - 1703 - 23 -	

ZAVRŠNI ZADATAK

Student: **Dora Baronica** JMBAG: **0035217025**

Naslov rada na hrvatskom jeziku: **Računalni model strojnog učenja za sortiranje otpada**

Naslov rada na engleskom jeziku: **Machine learning computational model for waste sorting**

Opis zadatka:

Modeli dubokog učenja mogu biti trenirani na velikom skupu podataka kako bi osigurali prepoznavanje i klasifikaciju objekata, različitih pojava, bića i sl. Rezultati zaključivanja modela se mogu koristiti kao temelji različitih aplikacija iz računalnog ili realnog svijeta. Tako je moguće trenirati i kreirati računalni model za sortiranje otpada koristeći računalni vid.

U radu je potrebno:

- proučiti metode dubokog učenja te odabrati prikladne za rješavanje zadanog problema s posebnim naglaskom na resnet34 mrežnu arhitekturu,
- pronaći, obraditi i organizirati odgovarajući skup slika koji će se koristiti za trening računalnog modela,
- trenirati model na zadanom skupu slika koristeći različite postavke učenja (engl. hyperparameters),
- evaluirati rad modela kroz eksperimente u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava te dati kritički osvrt.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:

30. 11. 2022.

Zadatak zadao:

Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:

- 1. rok: 20. 2. 2023.
- 2. rok (izvanredni): 10. 7. 2023.
- 3. rok: 18. 9. 2023.

Predviđeni datumi obrane:

- 1. rok: 27. 2. - 3. 3. 2023.
- 2. rok (izvanredni): 14. 7. 2023.
- 3. rok: 23. 9. - 29. 9. 2023.

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru izv. prof. dr. sc. Tomislavu Stipančiću na pruženoj pomoći i podršci tijekom izrade ovog rada, ali i na pruženom znanju kroz razne kolegije proteklih godina studiranja.

Zahvaljujem se svojoj obitelji i prijateljima na podršci i motivaciji tijekom cijelog studija.

Dora Baronica

Sadržaj

Sadržaj	I
Popis slika	II
SAŽETAK	III
SUMMARY	IV
1. Uvod	5
2. Što je umjetna inteligencija?	6
3. Duboko učenje	7
3.1. Metode dubokog učenja	8
3.1.1. Rekurentne neuronske mreže	9
3.1.2. Konvolucijske neuronske mreže	10
4. Python biblioteke	17
5. Obrada i organizacija podataka	18
5.1. Organizacija fotografija otpada	18
6. Obuka modela	21
6.1. ResNet-50	21
6.1.1. Obuka prethodno obučenog ResNet-50 modela	22
6.1.2. Augmentacija podataka	23
6.1.3. Obuka modela s proširenim podacima	25
7. Evaluacija modela	26
7.1. Metrike klasifikacije	26
7.1.1. Točnost (Accuracy)	27
7.1.2. Matrica konfuzije	30
8. Zaključak	32
Reference	33
Prilozi	34
Prilog I: Programski kod	35

Popis slika

Slika 1. Razlika strojnog učenja i dubokog učenja	7
Slika 2. Podjela metoda dubokog učenja	8
Slika 3. Rekurentna neuronska mreža	9
Slika 4. Slika kao mreža piksela	10
Slika 5. Konvolucijski sloj	11
Slika 6. Operacija udruživanja.....	11
Slika 7. Arhitektura VGG neuronske mreže.....	13
Slika 8. Problem nestajanja gradijenta (Sigmoid funkcija)	14
Slika 9. Usporedba rada VGG-19, obične neuronske mreže i ResNet-34 neuronske mreže.....	16
Slika 10. Izbor i uvoz biblioteka	17
Slika 11. Ekstrahiranje zip datoteke.....	18
Slika 12. Direktoriji skupova podataka	18
Slika 13. Stvaranje varijable i promjena veličine slika	19
Slika 14. Postavljanje podatkovnog cjevovoda	19
Slika 15. Arhitektura ResNet-50.....	22
Slika 16. Izgradnja modela dubokog učenja s prethodno obučeno ResNet-50 arhitekturom.....	22
Slika 17. Geometrijska transformacija	24
Slika 18. Transformacija boja.....	24
Slika 19. Kernel filter.....	24
Slika 20. Linija koda za augmentaciju podataka	25
Slika 21. Prikaz koda za obuku modela	25
Slika 22. Prikaz podjele skupova podataka	26
Slika 23. Dio koda za generiranje linijskog dijagrama koji prikazuje točnost	27
Slika 24. Dijagram točnosti	28
Slika 25. Dio koda za generiranje linijskog dijagrama koji prikazuje gubitak	29
Slika 26. Dijagram funkcije gubitka	29
Slika 27. Matrica konfuzije	30
Slika 28. Dio koda koji prikazuje stvaranje matrice konfuzije.....	31
Slika 29. Konačna matrica konfuzije	31

SAŽETAK

Globalni problem upravljanja smećem zahtijeva kreativne pristupe pojednostavljenju postupaka sortiranja. Moderni tokovi otpada predstavljaju jedinstvene izazove za konvencionalne pristupe, zbog čega je nužno korištenje najsuvremenijih tehnologija. Ovim radom cilj je prikazati mogućnosti korištenja potencijala strojnog učenja u svrhu pronalaženja novih strategija za pristup navedenoj problematici. Prikazat će se razvoj računalnog modela strojnog učenja za sortiranje otpada tako što će se trenirati već postojeća mrežna arhitektura u programskom jeziku Python uz korištenje potrebnih biblioteka. Prije samog procesa bilo je potrebno proučiti i odabrati adekvatne metode dubokog učenja prikladne za rješavanje zadanog problema. Isto tako, kako su temelj strojnog učenja odgovarajući skupovi podataka, bilo ih je potrebno pronaći, obraditi i organizirati. U ovom slučaju skup podataka predstavlja skup fotografija raznih vrsta otpada. Nakon svega, prikazat će se evaluacija rada modela kroz eksperimente u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava.

Ključne riječi: strojno učenje, duboko učenje, mrežna arhitektura, obuka modela, skupovi podataka, evaluacija, klasifikacija, umjetna inteligencija

SUMMARY

The global problem of waste management requires creative approaches to simplify sorting procedures. Modern waste streams present unique challenges to conventional approaches, which is why the use of state-of-the-art technologies is necessary. The aim of this paper is to show the possibilities of using the potential of machine learning in order to find new strategies for approaching the mentioned problem. The development of a machine learning computational model for waste sorting will be presented by training an already existing neural network in the Python programming language using the necessary libraries. Before the process itself, it was necessary to study and select adequate deep learning methods suitable for solving the given problem. Also, since the basis of machine learning is the appropriate datasets, it was necessary to find, process and organize them. In this case, the dataset represents a set of photographs of various types of waste. After all, the evaluation of the model's work through experiments as part of the Laboratory for Designing Manufactured and Assembled Systems will be presented.

Keywords: machine learning, deep learning, network architecture, model training, datasets, evaluation, classification, artificial intelligence

1. Uvod

Eksplozivna urbanizacija i rastuća globalna populacija obilježja su modernog doba. Povećano stvaranje otpada jedna je od loših posljedica ovih pojava te se tako postavlja pitanje kako što pravilnije zaštititi okoliš i živjeti održivo. Učinkovito upravljanje otpadom postalo je globalni izazov, a njegovo rješenje znanstvenici, ekolozi i inženjeri pokušavaju pronaći u vrhunskim i modernim tehnologijama razvijajući inovativna rješenja koja ne samo da ublažavaju negativne utjecaje otpada, već i promiču učinkovitost resursa.

Tradicionalne metode razvrstavanja otpada teško održavaju korak sa složenošću i obujmom tokova otpada te se javlja nužna potreba za istraživanjem naprednih tehnika koje mogu poboljšati točnost i učinkovitost procesa.

U posljednjem desetljeću, kroz razvoj tehnologije pojavilo se revolucionarna paradigma zvana duboko učenje, kao grana strojnog učenja, te potaknulo uzbuđenje i potencijal za inovaciju, u kontekstu navedene problematike, omogućavanjem automatizirane inteligentne klasifikacije različitih otpadnih materijala u stvarnom vremenu. Iskorištavanje složene arhitekture neuronskih mreža za razotkrivanje složenih obrazaca i prikaza iz neobrađenih podataka, čini ga neusporedivim alatom za rješavanje kompleksnih problema koji su se nekada smatrali nepremostivima.

Primarni cilj ovog rada je upravo na primjeru ovog globalnog ekološkog problema razviti računalni model koji korištenjem računalnog vida razvrstava otpad.

Prvi dio rada fokusirat će se na prolaženje kroz različite metode dubokog učenja i biranja prikladne navedenoj problematici. Kasnije je potrebno pronaći i obraditi odgovarajući skup podataka na kojem će se zatim model trenirati i njegov rad evaluirati.

2. Što je umjetna inteligencija?

Dobitnik Turingove nagrade Marvin Minsky, kaže da umjetna inteligencija stoji kao svjetionik ljudskog postignuća, predstavljajući naše kolektivno nastojanje da strojevima podarimo sposobnost oponašanja kognitivnih funkcija koje su se prije pripisivale isključivo ljudskom intelektu. [1]

Tehničko čudo koje nadilazi jednostavne algoritamske operacije stvoreno je kao rezultat ovog interdisciplinarnog napora, koji je spojio niti računalne znanosti, matematike, neuroznanosti i inženjerstva.

Nema mnogo znanstvenih otkrića u ljudskoj povijesti koja imaju toliko potencijala i zanimljivosti kao umjetna inteligencija. To je rezultat godina istraživanja kognitivnih sposobnosti koje karakteriziraju ljudski intelekt i koje je moguće oponašati, ako ne i premašiti ih. U osnovi, umjetna inteligencija je simfonija algoritama, podataka i računalne snage orkestrirane da nalikuju složenim postupcima ljudskog intelekta, percepcije i donošenja odluka, a pokušaj da se zatvori jaz između ljudske spoznaje i računalne sposobnosti je hrabar. Umjetna inteligencija, inspirirana idejama Alana Turinga o računalima koja oponašaju ljudsku misao, ubrznim se korakom razvila u veliko područje niza metodologija koje uključuju, primjerice, strojno učenje i obradu prirodnog jezika.

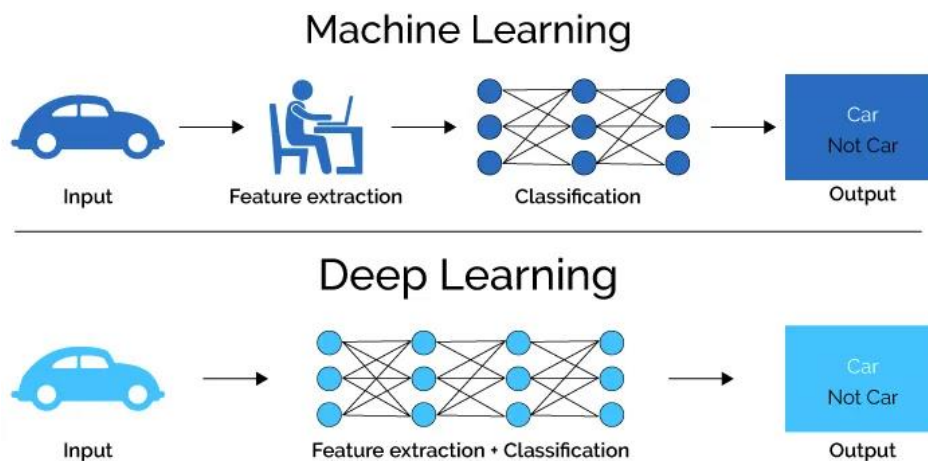
Evolucijska putanja umjetne inteligencije prati zadivljujuće putovanje od rudimentarnih simboličkih sustava razmišljanja do današnjeg doba kojim dominiraju strojno učenje i zamršene neuronske mreže. Kako navedene tehnologije napreduju, pomiču granice onoga za što su strojevi sposobni, donoseći dramatične promjene u nizu sektora, uključujući zdravstvenu skrb, financije, prijevoz itd. Osim svojih korisnih primjena, AI promiče etičku i filozofsku raspravu, nadahnjujući razmatranja o svijesti, autonomiji i samoj prirodi inteligencije.

3. Duboko učenje

Duboko učenje je podskup strojnog učenja, koje je u biti neuronska mreža s tri ili više slojeva. Ove neuronske mreže pokušavaju simulirati ponašanje ljudskog mozga, iako je to daleko od njegove sposobnosti, omogućujući mu da "uči" iz velikih količina podataka. Premda neuronska mreža s jednim slojem još uvijek može dati približna predviđanja, dodatni skriveni slojevi mogu pomoći u optimizaciji i poboljšanju točnosti. [2]

Ovom definicijom dubokog učenja jasno je naglašeno da duboko učenje nije isto što i strojno učenje već samo njegov podskup što je bitno znati. Jedna od glavnih razlika ove dvije metode je ta što modeli strojnog učenja iako pri preuzimanju novih podataka postaju bolji u obavljanju svojih funkcija, i dalje im je ponekad potrebna ljudska intervencija. Inženjeri su potrebni kako bi napravili prilagodbe ukoliko algoritam umjetne inteligencije vrati netočno predviđanje.

Model dubokog učenja, s druge strane, pruža mogućnost da algoritam sam određuje točno ili netočno predviđanje bez pomoći čovjeka.



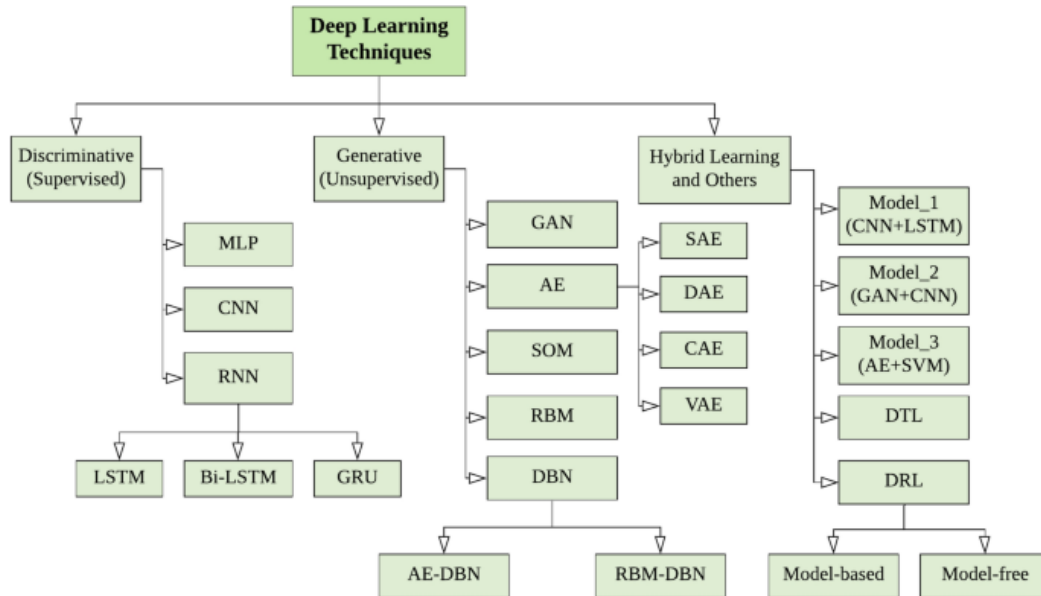
Slika 1. Razlika strojnog učenja i dubokog učenja

Također, ključna razlika je i u tome što je Duboko učenje općenito složenije te zahtijeva veći skup podataka kako bi se dobili pouzdani rezultati. Iz toga proizlaze zahtjevi za posjedovanjem upravo velikih količina tih podataka i GPU visokih performansi kako bi se spomenuti mogli obraditi.

Duboko učenje obuhvaća niz metoda i tehnika koje se koriste za obuku i implementaciju dubokih neuronskih mreža za različite zadatke te ih je bitno proučiti i odabrati prikladnu za rješavanje danog problema.

3.1. Metode dubokog učenja

Metode odnosno tehnike dubokog učenja mogu se najprije podijeliti na tri skupine: generativno, hibridno i diskriminativno duboko učenje. Svaka od pojedinih skupina ima i svoje podskupine, a potpunu klasifikaciju najlakše je prikazati slikovno [Slika 2.].



Slika 2. Podjela metoda dubokog učenja

Generativni model dubokog učenja (bez nadzora) općenito se može definirati na sljedeći način: Generativni model opisuje kako se skup podataka generira, u smislu probabilističkog modela. Uzorkovanjem iz ovog modela možemo generirati nove podatke. [3]

Kao što je navedeno u samoj definiciji, cilj ovog modela je generirati nove skupove značajki koje izgledaju kao da su stvorene korištenjem istih pravila kao izvorni podatci.

Recimo da je primjer entiteta koji se želi generirati slika predmeta, potreban je skup podataka od mnogo slika tog predmeta a značajke su u ovom slučaju pojedinačne vrijednosti piksela. Skup tih podataka poznati su kao podatci za obuku. Generativni model također mora biti probabilistički, a ne deterministički. Ako je naš model samo fiksni izračun, kao što je uzimanje prosječne vrijednosti svakog piksela u skupu podataka, on nije generativan jer model daje isti rezultat svaki put. Model mora uključivati stohastički (slučajni) element koji utječe na pojedinačne uzorke koje model generira.[3]

Diskriminativni modeli, s druge strane, nemaju za cilj generirati nove skupne podatka već na temelju određenog skupa podataka prepoznati pripada li novo dani podatak toj skupini.

Treniramo li model, primjerice, na fotografijama određenog predmeta, naučio bi da je dani predmet određenih oblika, tekstura i boja te razlikovao iste od oblika, tekstura i boja na testnim fotografijama.

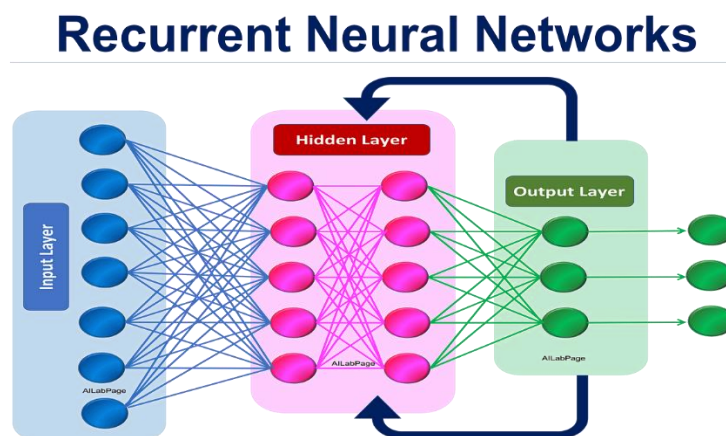
Drugim riječima, diskriminativno modeliranje pokušava procijeniti vjerojatnost da opažanje x pripada kategoriji y . Generativno modeliranje ne mari za označavanje opažanja. Umjesto toga, pokušava procijeniti vjerojatnost da se opažanje uopće vidi. [3]

3.1.1. Rekurentne neuronske mreže

Rekurentne neuronske mreže (RNN) jedan su od temelja inovacija u dinamičnom svijetu umjetne inteligencije, gdje potrazi za razumijevanjem i modeliranjem kompliciranih vremenskih korelacija nema kraja.

Ovi izvanredni računalni modeli potpuno su promijenili način na koji se vrši sekvencijalna obrada podataka i kao takvi su posebno korisni u područjima kao što su prepoznavanje govora, analiza vremenskih nizova i obrada prirodnog jezika.

Princip rada rekurentne neuronske mreže razlikuje se od principa rada tradicionalnih neuronskih mreža po tome što su u tradicionalnim neuronskim mrežama svi ulazi i izlazi neovisni jedni o drugima. S druge strane, u slučajevima kada je, primjerice, potrebno predvidjeti sljedeću riječ rečenice, potrebne su prethodne riječi te ih je stoga potrebno zapamtiti. Upravo je tako nastala rekurentna neuronska mreža gdje je ovaj problem riješen uz pomoć skrivenog sloja [Slika 3.]. Glavna i najvažnija značajka RNN-a je njegovo skriveno stanje, koje pamti neke informacije o nizu. Stanje se također naziva i stanje memorije budući da pamti prethodni unos u mrežu. Koristi iste parametre za svaki ulaz jer obavlja isti zadatak na svim ulazima ili skrivenim slojevima za proizvodnju izlaza. Time se smanjuje složenost parametara, za razliku od drugih neuronskih mreža. [4]



Slika 3. Rekurentna neuronska mreža

Rekurentne neuronske mreže vrlo su moćan alat za rad te se široko primjenjuju u svakodnevnom životu. Zbog napretka u arhitekturi modela, algoritama za obuku i paralelnog računanja tijekom proteklih trideset godina, prestale su biti modeli primarno od interesa za kognitivno modeliranje i računalnu neuroznanost

snažni i praktični alati za opsežno nadzirano učenje iz sekvenci. [5]

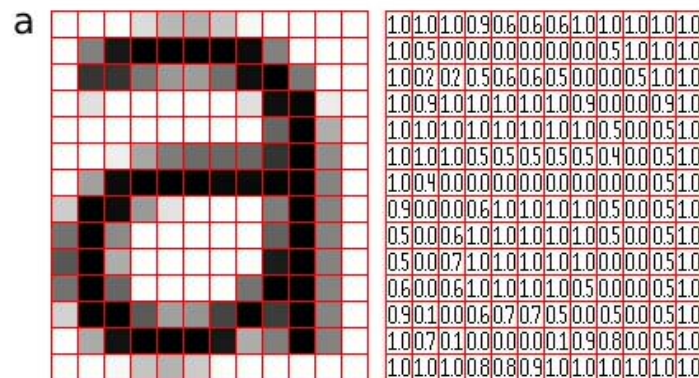
Izazovi u njihovoj primjeni ipak nisu neizbježni pa računalni troškovi obuke RNN-ova mogu biti visoki, osobito kada se radi s ogromnim skupovima podataka i zamršenim mrežnim dizajnom. Njihova sposobnost skaliranja i raspon upotrebe bi tako mogao biti ograničen. [6]

3.1.2. Konvolucijske neuronske mreže

Jedinstvena vrsta algoritama koji, u golemom području umjetne inteligencije, imaju nevjerojatnu sposobnost da percipiraju svijet poput ljudi nazivaju se konvolucijske neuronske mreže (CNN). Može ih se zamisliti kao digitalne umjetnike koji vole uzorke, rubove i oblike.

Konvolucijske neuronske mreže pažljivo ispituju slike i skeniraju ih piksel po piksel. Ne fokusiraju se na veliku sliku već ulaze u duboko u detalje kako bi pronašle suptilnosti koje bi nevještom oku promaknule.

Digitalna slika je binarni prikaz vizualnih podataka i sadrži upravo niz piksela raspoređenih u obliku mreže, prikazane na slici [Slika 4.], čiji dijelovi sadrže vrijednosti tih piksela koje označavaju koliko bi svaki piksel trebao biti svijetao i koje boje. [7]



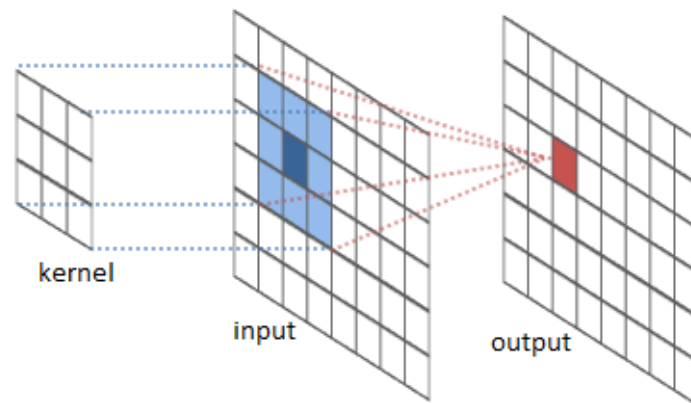
Slika 4. Slika kao mreža piksela

Slojevi konvolucijske neuronske mreže raspoređeni su tako da najprije otkrivaju jednostavnije uzorke poput linija i krivulja, gotovo kao da se slika skicira od nule. Zatim u sljedećem koraku prelaze na složenije detalje kao što su lica i objekti. Prepoznaju oblike unutar oblika, teksture unutar tekstura, sve dok ne izgrade višedimenzionalno razumijevanje onoga što je predmet proučavanja.

Konvolucijska neuronska mreža obično se sastoji od tri sloja: konvolucijski spoj, skupni sloj i potpuno povezani sloj.

Konvolucijski sloj ključni je i temeljni spoj blok CNN-a te nosi glavni dio računalnog opterećenja mreže. Može ga se zamisliti poput posebnog filtra koji skenira sliku skupljajući uzorke i detalje. Ovaj sloj uključuje operaciju množenja matrica gdje jedna matrica, nazvana kernel, stupa u interakciju s određenim dijelom ulaza, poznatim kao receptivno polje. Kernel je manje prostorne veličine, ali

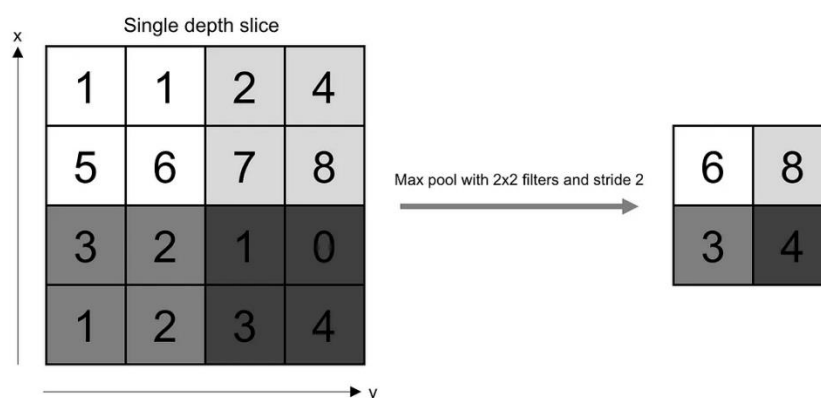
obuhvaća dubinu unosa. Tijekom prolaska naprijed, kernel klizi po visini i širini slike stvarajući slikovnu reprezentaciju tog receptivnog područja. Ovo proizvodi dvodimenzionalni prikaz slike poznat kao aktivacijska mapa koja daje odgovor kernela na svakom prostornom položaju slike. Klizna veličina jezgre naziva se korak. [7]



Slika 5. Konvolucijski sloj

Skupni sloj ili sloj udruživanja zamjenjuje izlaz mreže na određenim lokacijama izvođenjem sažete statistike obližnjih izlaza. Ova operacija obrađuje se na svakom odsječku pojedinačno te pomaže smanjenju prostorne veličine prikaza, što smanjuje potrebnu količinu izračuna i težine. [7]

Dostupno je nekoliko algoritama za udruživanje, uključujući L2 normu pravokutnog susjedstva, prosjek pravokutnog susjedstva i ponderirani prosjek temeljen na udaljenosti od središnjeg piksela. Ipak, najpopularniji proces je maksimalno udruživanje, prikazan na slici [Slika 6.], koje izvještava o maksimalnom izlazu iz susjedstva.



Slika 6. Operacija udruživanja

U potpuno povezanom sloju neuroni imaju punu povezanost sa svim neuronima u onom prethodnom i sljedećem. Ovo daje mogućnost izračuna matričnim množenjem nakon čega slijedi učinak pristranosti. Ovaj sloj pomaže mapirati povezanost između ulaza i izlaza.

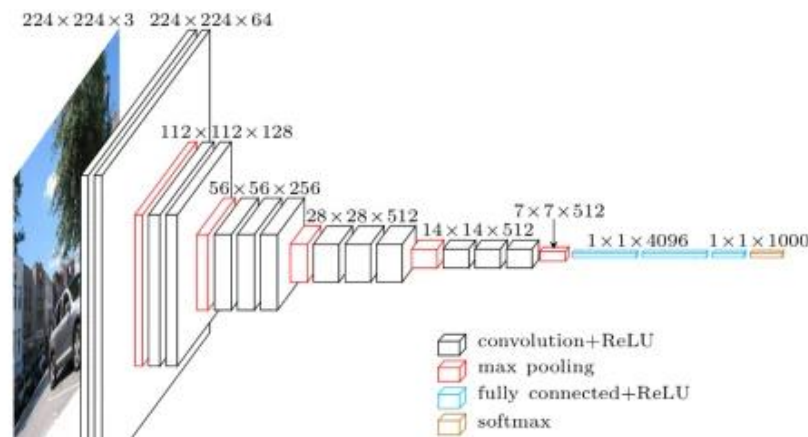
Cjelokupna struktura i raspored navedenih slojeva, uključujući njihovu vrstu i broj te njihove veze i protok informacija između njih, definiraju arhitekturu neuronske mreže. Izbor arhitekture neuronske mreže značajno utječe na njenu sposobnost da uči iz podataka. Dobro dizajnirana arhitektura može poboljšati izvedbu modela, dok loše odabrana arhitektura može uzrokovati nedovoljno prilagođavanje (model ne uspijeva uhvatiti temeljne uzorke u podacima) ili pretjerano prilagođavanje (model predobro uči podatke o obuci, ali se muči s novim, nepoznati podaci). Ovisi o prirodi zadatka i vrsti podataka potrebni su različiti dizajni odnosno arhitekture mreža, a neke od popularnih CNN arhitekture uključuju LeNet, AlexNet, VGG, ResNet i Inception, svaka sa svojom jedinstvenom filozofijom dizajna i karakteristikama izvedbe.

Konvolucijske neuronske mreže revolucionirale su prepoznavanje uzoraka i računalni vid. Slojevi mreže čine osebujni dizajn koji im omogućuje uspješno učenje hijerarhijskih značajki iz slika. Ovim postupkom pokazale su se iznimno dobrima u zadacima kao što su segmentacija, identifikacija objekata i kategorizacija slika. Osim toga, konvolucijska neuronska mreža pokazala je otpornost na promjene u položaju objekta i osvjetljenju što ih čini nevjerojatno prilagodljivima u stvarnim aplikacijama. Postaju fleksibilnije za nove domene i skupove podataka kada se koriste unaprijed obučeni modeli, prijenos učenja i pristupi poput povećanja podataka te će se koristiti i u ovom radu za svrhu rješavanja problema klasifikacije otpada.

VGG (Visual Geometry Group)

VGG ili VGGNet standardna je arhitektura duboke konvolucijske neuronske mreže s više slojeva. Pridjev „duboke“ ispred spomenutih konvolucijskih mreža označava i odnosi se na broj slojeva kod VGG-16 i VGG-19 arhitekture koje se sastoje upravo od njih 16 odnosno 19. [8]

VGG dizajn definiran je pravilnošću i jednostavnošću. Sastoji se od niza konvolucijskih slojeva sa sićušnim 3x3 filtrima, nakon kojih uvijek slijede slojevi koji maksimiziraju udruživanje, prikazano na slici [Slika 7.].



Slika 7. Arhitektura VGG neuronske mreže

Upravo je zahvaljujući ovom dosljednom dizajnu mreža postigla izvanredne performanse u prepoznavanju objekata koji se. Unatoč tome što je računalno zahtjevan, pokazao se nevjerojatno učinkovitim u izdvajanju složenih karakteristika iz fotografija.

Kao što je gore spomenuto, VGGNet-16, primjerice, podržava 16 slojeva i može klasificirati fotografije u 1000 kategorija objekata, uključujući tipkovnicu, životinje, olovku, miša itd te ima ulaznu veličinu slike od 224 x 224. [8]

Model VGG16 postiže gotovo 92,7% top-5 točnosti testa u ImageNetu, skupu podataka koji se sastoji od više od 14 milijuna slika koje pripadaju gotovo 1000 klasa. Također, bio je to jedan od najpopularnijih modela predanih na ILSVRC-2014. [8]

A. Zisserman i K. Simonyan sa Sveučilišta u Oxfordu predložili su VGG16 model 2014. godine te ga objavili u istraživačkom radu pod naslovom "Vrlo duboke konvolucijske mreže za prepoznavanje velikih slika" .

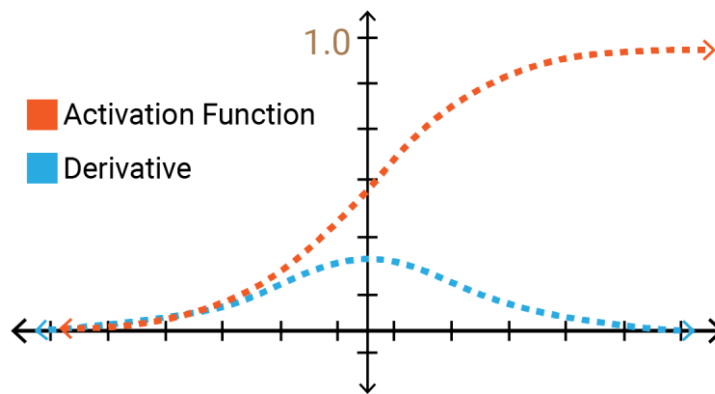
Sve u svemu, mreže VGG pokazale su izvanrednu učinkovitost u izvlačenju složenih informacija iz fotografija unatoč visokim zahtjevima za obradu. Iako su se pojavile novije arhitekture, VGG je i dalje kamen temeljac i standard u stvaranju konvolucijskih neuronskih mreža (CNN). Utjecaj u primjeni naglašava koliko su dobro strukturirane arhitekture važne za proširenje mogućnosti zadataka obrade slike, a istraživači koji rade na unapređenju tehnologije računalnog vida neprestano su motivirani i vođeni nasljeđem VGG-a.

ResNet (ResNet-34)

Rezidualna mreža (Residual Network), od čega dolazi skraćenica ResNet, inovativne je neuronska mreža koju su 2015. godine predstavili Kaiming He, Xiangyu Zhang, Shaoqing Ren i Jian Sun u svom istraživačkom radu tematike računalnog vida pod imenom „Duboko rezidualno učenje za prepoznavanje slike“. [9]

Ovaj model, također 2015. godine, osvojio je prvo mjesto na klasifikacijskom natjecanju ILSVRC, s pogreškom od samo 3.57 % čime dokazuje svoju iznimnu uspješnost. [9]

Rezidualna mreža postavljena je kao rješenje problema nestajanja gradijenta u iznimno dubokim neuronskim mrežama. Problem nestajanja gradijenta javlja se kod dodavanja više slojeva koji koriste određene aktivacijske funkcije. [10] Aktivacijske funkcije matematičke su operacije koje se primjenjuju na svaki neuron mreže te pomažu određivanju izlaza tog neurona s obzirom na skup ulaza. Svrha aktivacijske funkcije jest uvesti nelinearnost u mrežu omogućujući joj da aproksimira i uči složene nelinearne odnose u podacima. One ipak mogu stvarati probleme kod dubokih mreža kako je prikazano na slici [Slika 8.], na primjeru Sigmoid aktivacijske funkcije.



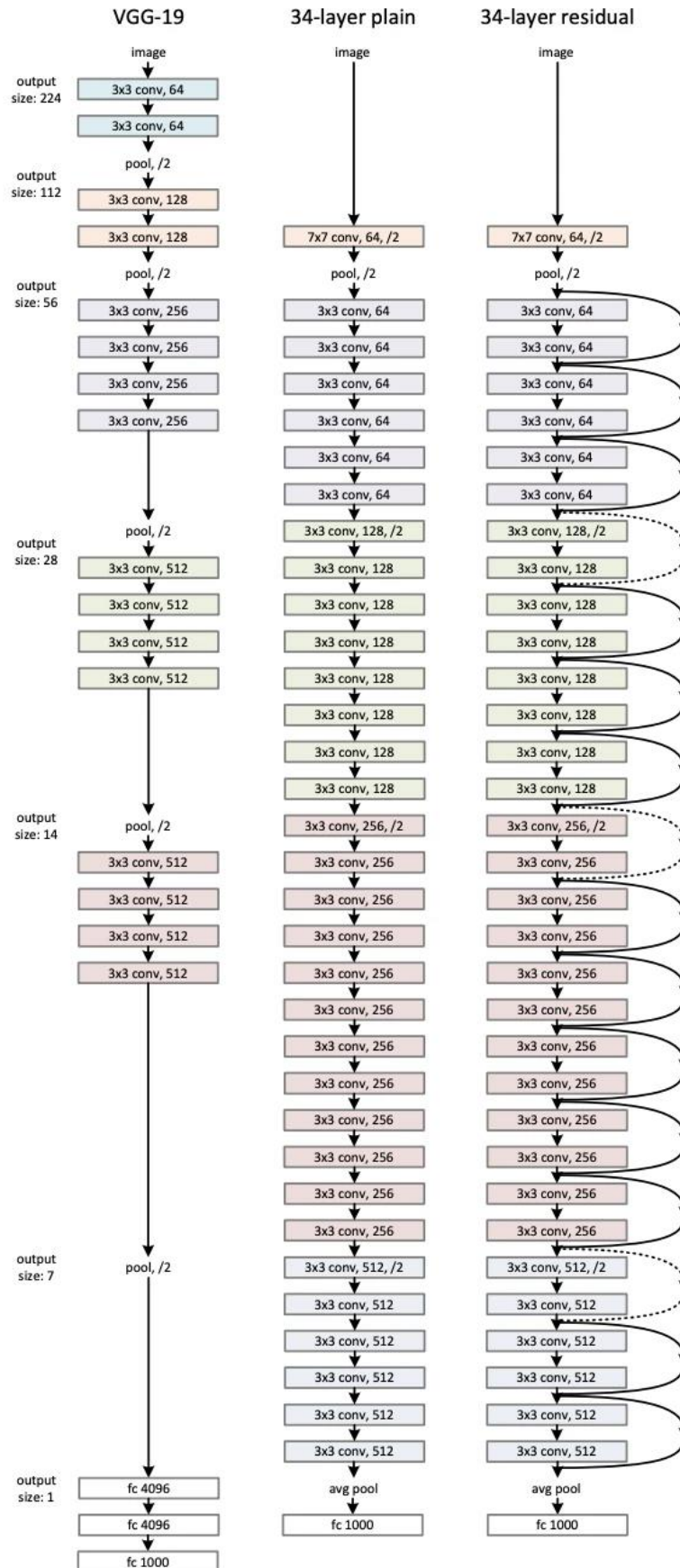
Slika 8. Problem nestajanja gradijenta (Sigmoid funkcija)

Kako je vidljivo, derivacije aktivacijske funkcije postaje vrlo mala (približava se nuli), a ona utječe na to koliko se težine u određenom sloju prilagođavaju tijekom treninga. Kada gradijent postane iznimno mali težine ranijih slojeva u mreži ne ažuriraju se učinkovito te oni mogu završiti tako da uopće ne nauče puno.

ResNet razvijen je dakle upravo s namjerom da riješi ovaj problem. Snaga ove vrste neuronske mreže jest ideja "preskakanja veza", koja je temelj rezidualnih blokova. Ove veze za preskakanje rade na dva načina. Najprije ublažavaju problem nestajanja gradijenta postavljanjem alternativnog prečaca kroz koji gradijent prolazi. Osim toga, omogućuju modelu da nauči funkciju identiteta. Time se osigurava da viši slojevi modela rade jednako dobro kao i oni niži. Kao rezultat toga, ResNet poboljšava učinkovitost dubokih neuronskih mreža s više neuronskih slojeva dok u isto vrijeme minimalizira postotak pogrešaka. [9]

Prva ResNet arhitektura bila je Resnet-34 prikazana na slici [Slika 9.] u usporedbi s VGG neuronskom mrežom i običnom mrežom. 34- slojna rezidualna mreža može se zamisliti poput slganja tornja blokova. Rastom toga tornja on postaje sve nestabilniji. ResNet34 stvara male prečace ili mostove između blokova što se može usporediti s povremenim dodavanjem snažne potporne konstrukcije u sredini tornja. Navedeni postupak omogućuje nastavak gradnje bez previše brige da će se urušiti. Zbog toga su ResNet34 i slične arhitekture bile revolucionarne u dubokom učenju. Omogućile su izgradnju

nevjerojatno duboke mreže, koje su sposobne razumjeti vrlo složene obrasce u u predmetima poput slika što ih čini moćnim alatima u rješavanju zadataka takve tematike.



Slika 9. Usporedba rada VGG-19, obične neuronske mreže i ResNet-34 neuronske mreže

4. Python biblioteke

Odabir biblioteka, prikazanih na slici [Slika 10.], kod izrade programa za klasifikaciju otpada, kao i kod svakog drugog programa, snažan je skup alata dizajniran za stvaranje i procjenu modela strojnog učenja. *NumPy* (*np*) olakšava izvođenje vitalnih numeričkih izračuna i manipulacija nizovima koji su potrebni za operacije u linearnoj algebri, ključnom elementu mnogih tehnika strojnog učenja. Googleov okvir za strojno učenje otvorenog koda, *TensorFlow*, pruža robustan skup alata za stvaranje i obuku različitih vrsta neuronskih mreža dok je *Keras* poboljšanje *TensorFlowa*. Arhitektura neuronske mreže može se definirati kombiniranjem *Layers* i *Dense* modula iz *Kerasa* s njihovim *TensorFlow* ekvivalentima. Linearni skup slojeva može se stvoriti korištenjem sekvencijalnog tipa modela iz *Kerasa*, što je prikladna metoda za mnoge aplikacije dubokog učenja. Kao snažan alat za minimiziranje funkcije gubitka tijekom treninga, Adam, funkcija optimizacije automatski prilagođava stope učenja za svaki parametar.

Matplotlib.pyplot i *Seaborn* surađuju kako bi ponudili fleksibilne alate za izradu grafikona, dijagrama i vizualnih prikaza performansi modela na frontu vizualizacije. Uključivanje matrice konfuzije, iz modula *sklearn.metrics*, koja nudi uvid u stvarne pozitivne, istinske negativne, lažno pozitivne i lažno negativne rezultate, nudi sofisticiranu procjenu izvedbe algoritma klasifikacije. Zajedno, ove knjižnice stvaraju kohezivnu zbirku resursa koja omogućuje izgradnju, obuku i procjenu modela dubinskog učenja i stjecanje temeljitog znanja o njihovoj izvedbi.

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras import layers
from tensorflow.python.keras.layers import Dense, Flatten
from keras.models import Sequential
from keras.optimizers import Adam
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

Slika 10. Izbor i uvoz biblioteka

5. Obrada i organizacija podataka

Usredotočenost na razumijevanje podataka, njihovu važnost i ulogu jednako je bitna kao i usredotočenost na izgradnju modela strojnog učenja i rješavanje problema. Danas postoje velike količine dostupnih skupova podataka no nedostatak onih kvalitetnih i kvantitativnih je također prisutan. U području umjetne inteligencije i strojnog učenja upravo zato postoji poslovice „smeće unutra, smeće van“ koja naglašava bitnost kvalitete podataka na kojima se zatim neuronske mreže treniraju, podešavaju i ocjenjuju.

Priprema i razumijevanje podataka jedan je od najvažnijih i najdugotrajnijih zadataka životnog ciklusa projekta strojnog učenja. Istraživanje pokazuje da većina podatkovnih znanstvenika i programera koji se bave umjetnom inteligencijom troše gotovo 70% svog vremena analizirajući skupove podataka. Preostalo vrijeme troši se na druge procese kao što su odabir modela, obuka, testiranje i implementacija. [11]

Potrebe problematike ovog rada zadovoljit će skup podataka u obliku fotografija raznih vrsta otpada smještenih u zip datoteku. [12]

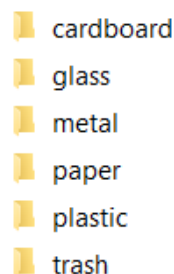
5.1. Organizacija fotografija otpada

Prvi korak programa bit će ekstradiranje navedene datoteke prikazano na slici [Slika 11.] kako bi se dobilo šest direktorija koji sadrže različite vrste otpada [Slika 12.].

```
import zipfile as zf

files = zf.ZipFile("dataset-resized.zip", 'r')
files.extractall()
files.close()
```

Slika 11. Ekstradiranje zip datoteke



Slika 12. Direktoriji skupova podataka

U sljedećem koraku, nakon ekstrahiranja podataka, direktorij *data*, u kojemu su podatci pohranjeni, spremiće se u varijablu *data_dir*, ako je prikazano na slici [Slika 13.], kako bi se na taj način mogao koristiti dalje u kodu.

```
data_dir = ("data")

img_height, img_width = 180, 180
batch_size = 32
```

Slika 13. Stvaranje varijable i promjena veličine slika

Također, kako je prikazano na slici [Slika 13.], potrebno je i promijeniti veličine slika iz skupa na jedinstvenu veličinu. Ovo je ključan korak kod ubacivanja fotografija u neuronsku mrežu jer osigurava da svaka slika ima istu veličinu, što je bitno da bi je mreža pravilno obradila. Varijabla *batch_size* kontrolira koliko se fotografija obrađuje tijekom obuke u jednoj iteraciji. Model će, u ovom slučaju, obraditi 32 fotografije odjednom prije ažuriranja svojih unutarnjih parametara. Ovo je tipična strategija dubokog učenja budući da omogućuje učinkovitije računanje i može ubrzati proces obuke. Količina dostupne memorije, složenost modela i veličina skupa podataka mogu utjecati na odabranu veličinu serije.

Sljedeći korak bit će postavljanje podatkovnog cjevovoda (tzv. Data pipeline) za obuku neuronske mreže pomoću *TensorFlowa* kako je prikazano na slici [Slika 14.].

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size)

class_names = train_ds.class_names
```

Slika 14. Postavljanje podatkovnog cjevovoda

Fotografije iz određenog direktorija (*data_dir*) čitau se i odvajaju u dva skupa podataka, za obuku i za provjeru valjanosti algoritma (*train_ds* i *val_ds*). 20% podataka će se koristiti za validaciju, a preostalih

80% će se koristiti za obuku, prema *validation_split=0.2* opciji. Ovi skupovi igraju različite uloge u razvoju, evaluaciji i finom podešavanju modela dubinskog učenja, osiguravajući robusnu izvedbu i mogućnost generalizacije.

Skup podataka za obuku zaslužan je za podučavanje neuronske mreže temeljnim obrascima, značajkama i odnosima svojstvenim podacima.

Dio podataka za provjeru valjanosti je Uzorak podataka korišten za pružanje nepristrane procjene modela koji odgovara skupu podataka za obuku tijekom podešavanja hiperparametara modela. Procjena postaje pristranija kako se vještina o validacijskom skupu podataka uključuje u konfiguraciju modela. [4]

Seed=123 osigurava ponovljivost, održavajući dosljednu randomizaciju. Veličina slika se mijenja na određene dimenzije (*img_height* i *img_width*), a broj uzoraka u svakoj seriji određen je već objašnjenim *batch_size*. Na kraju, nazivi klasa izvlače se iz skupa podataka za obuku za referencu u sljedećim koracima. Ovaj je proces temeljan u pripremi podataka za obuku modela strojnog učenja.

6. Obuka modela

Ključna faza u stvaranju modela dubinskog i strojnog učenja je obuka modela. Kako bi model naučio i prilagodio svoje unutarnje parametre za generiranje preciznih predviđanja ili klasifikacija, mora mu biti izložena znatna količina podataka. Ovaj je postupak može se usporediti s učenikom koji uči iz udžbenika kako bi u potpunosti razumio predmet.

Model zatim iterativno poboljšava svoje interne prikaze tijekom obuke na temelju znanja koje pružaju podaci o obuci. Navedeni proces postiže se minimiziranjem određenog cilja, često poznatog kao funkcija gubitka, koja mjeri neslaganje između predviđanja modela i stvarne istine. Ako se prvi korak mogao usporediti s učenikom i njegovim učenjem predmeta, ovaj korak može se usporediti s učiteljem koji procjenjuje učenikove odgovore i usmjerava ga prema dubljem razumijevanju. Krajnji cilj obuke je omogućiti modelu da prenese ono što je naučio na nove, neistražene podatke.

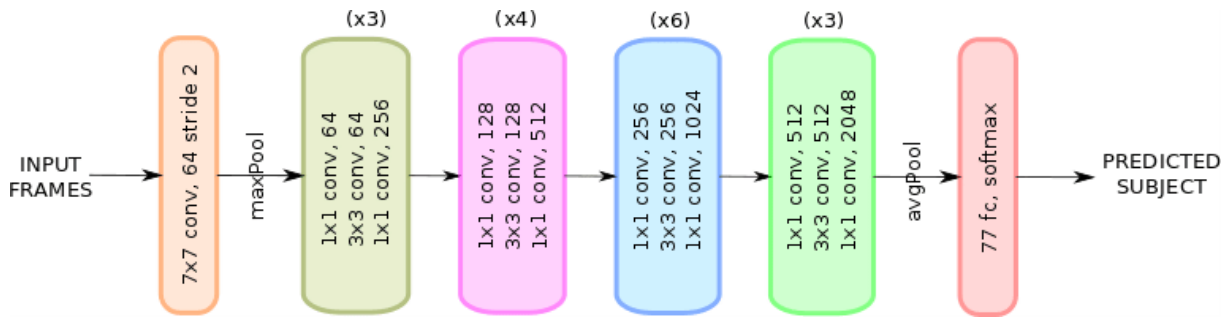
Model otkriva temeljne korelacije i obrasce u podacima o obučavanju, što mu omogućuje precizna predviđanja na prethodno neopaženim uzorcima.

Sve u svemu, obuka modela je temeljni postupak koji modele strojnog učenja osposobljava za obavljanje zadataka u rasponu od autonomnog donošenja odluka do identifikacije slike i obrade prirodnog jezika. Služi kao katalizator za pretvaranje neobrađenih podataka u snažan, inteligentan sustav.

6.1. ResNet-50

Rezidualna neuronska mreža ResNet-50 produžetak je arhitekture ResNet-34 i koristit će se u sklopu ovog rada za rješavanje problema klasifikacije otpada. Nadovezujući se na temelje koje je postavio ResNet-34, koji je već pokazao snagu zaostalih veza, ResNet-50 ovaj koncept produbljuje uvođenjem dodatnih slojeva i složenosti. Sastoji se od ukupno 50 slojeva te pokazuje povećanu sposobnost hvatanja zamršenih značajki i uzoraka u podacima.

Iako se arhitektura ResNet-50 temelji na modelu prikazanom na slici iz poglavlja 3.1.2. [Slika 9.], postoji jedna ključna razlika. U slučaju ResNet-50 arhitekture, građevinski blok je modificiran u dizajn uskog grla zbog zabrinutosti oko vremena potrebnog za obuku slojeva. Ovo je koristilo hrpu od 3 sloja umjesto ranija 2. Stoga je svaki od 2-slojnih blokova u Resnetu34 zamijenjen 3-slojnim blokom uskog grla, tvoreći Resnet 50 arhitekturu. Ovakav pristup, prikazan na slici [Slika 15.] ima znatno veću točnost od 34-slojnog ResNet modela. [9]



Slika 15. Arhitektura ResNet-50

6.1.1. Obuka prethodno obučenog ResNet-50 modela

U isječku koda prikazanom na slici [Slika 16.] prikazuje se kako izgraditi model dubokog učenja s prethodno obučenom ResNet-50 mrežnom arhitekturom u svrhu obavljanja zadatka klasifikacije otpada.

```
resnet_model = Sequential()

pretrained_model = tf.keras.applications.ResNet50(include_top=False,
                                                input_shape=(180, 180, 3),
                                                pooling='avg', classes=6,
                                                weights='imagenet')

for layer in pretrained_model.layers:
    layer.trainable = False

resnet_model.add(pretrained_model)
resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation='relu'))
resnet_model.add(Dense(6, activation='softmax'))

resnet_model.summary()

resnet_model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Slika 16. Izgradnja modela dubokog učenja s prethodno obučeno ResNet-50 arhitekturom

Inicijalizacija sekvencijalnog modela (`resnet_model = Sequential()`), koji simulira linearni niz slojeva, prvi je korak u postupku. Sljedeći je korak konstruirati ResNet-50 model s onemogućenim gornjim potpuno povezanim slojevima. Onemogućavanje gornjih potpuno povezanih slojeva vrši se kod prenamjene unaprijed uvježbanog modela vlastiti, prilagođeni zadatak. Poanta je zadržavanja nižih slojeva koji su naučili općenitije značajke poput rubova, tekstura i osnovnih oblika jer se ove značajke mogu prenijeti na širok raspon vizualnih zadataka. Ovaj proces poznat je kao prijenos učenja i moćna je tehnika za postizanje visokih performansi s ograničenim podatcima o obuci.

U ovom koraku također se definira ulazni oblik (*input_shape*) od 180 piksela u visinu i 180 piksela u širinu te tri RGB kanala slike. Kako bi se ekstrakciji značajki dao čvrst temelj, ovaj se model inicijalizira korištenjem težina koje su već uvježbane na skupu podataka ImageNet.

Sljedeća petlja kroz slojeve prethodno uvježbanog modela označava ih kao nesposobne za treniranje čime se osigurava da njihove težine ostanu konstantne tijekom treninga. Sloj izravnavanja (*Flatten()*) dodaje se sekvencijalnom modelu nakon prethodno obučenog ResNet50 modela kako bi se prebacili s konvolucijskih na čvrsto povezane slojeve. Konačna klasifikacijska glava sastoji se od dva gusta sloja sa *softmax* i *ReLU* aktivacijskim funkcijama. Informacije o sloju i parametri koji se mogu obući daju se zajedno s pregledom arhitekture modela. Zatim se daje rijetka kategorička funkcija unakrsnog entropijskog gubitka za višeklasnu klasifikaciju, a Adamov optimizator koristi se za konstrukciju modela sa stopom učenja od 0,001.

U kontekstu strojnog učenja i dubinskog učenja, optimizacijski algoritam je strategija ili pristup koji se koristi za modificiranje parametara modela kako bi se minimizirala (ili maksimizirala) određena ciljna funkcija. Funkcija cilja ocjenjuje koliko dobro model obavlja određeni zadatak, poput klasificiranja podataka ili predviđanja budućnosti. Pronalaženje skupa vrijednosti parametara koji daje najnižu (ili najvišu) vrijednost funkcije cilja cilj je optimizacije. Izvedba i ponašanje modela određeni su ovim vrijednostima parametara. Adam, što je kratica za Adaptive Moment Estimation, popularan je optimizacijski pristup za obuku modela dubokog učenja. Kombinira prednosti RMSprop-a (Propagacija korijena srednjeg kvadrata) s momentom, dvije dodatne dobro poznate tehnike optimizacije.

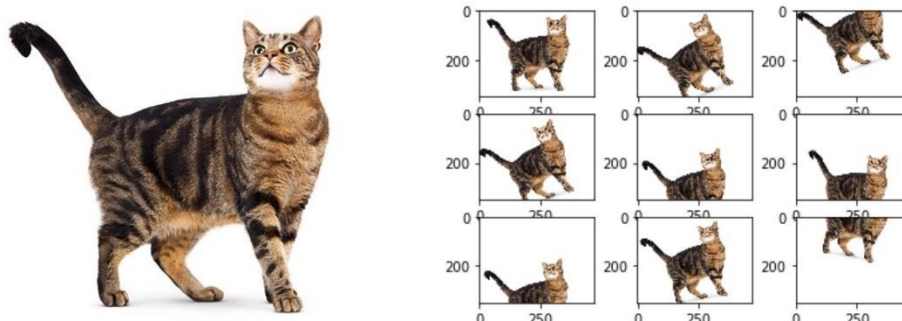
Stopa ili brzina učenja (*learning_rate*), koja u ovom slučaju iznosi 0,001, je hiperparametar koji određuje veličinu koraka pri kojoj se parametri modela ažuriraju tijekom obuke. Što je stopa učenja veća dolazi do brže kovergencije, ali ona isto tako može dovesti do rizika prekoračenja optimalnih vrijednosti. Nasuprot tome, niža stopa učenja rezultira manjim koracima koji mogu konvergirati sporije, ali s većom preciznošću. Ona dakle kontrolira veličinu prilagodbi učinjenih na težinama modela na temelju izračunatih gradijenata.

Važno je pravilno odabrati odgovarajuće stope učenja kako bi obuka modela bila uspješna. Ovaj korak često zahtijeva pažljivo podešavanje na temelju specifičnog skupa podataka i arhitekture modela.

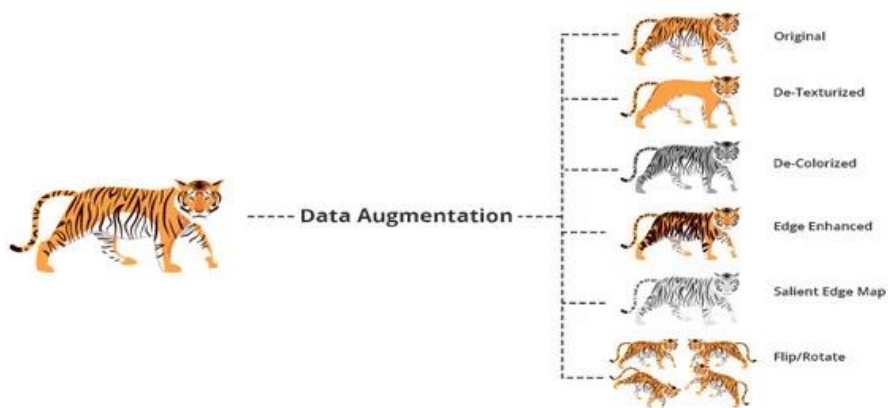
6.1.2. Augmentacija podataka

Područja računalnog vida i strojnog učenja uvelike se oslanjaju na koncept augmentacije odnosno povećanja podataka. Ono podrazumijeva izvođenje transformacija nad postojećim podacima, fotografijama, uključujući rotacije, okretanja, pomake, zumiranja [Slika 17.], transformacija boja [Slika 18.] te izoštravanje ili zamućivanje [Slika 19.]. Budući da je model tijekom ovog procesa izložen većem broju poboljšanih slučajeva, dobiva povećanu robusnost i prilagodljivost. Povećanje podataka u biti pomaže sposobnosti modela da generalizira nepredviđene promjene u ulaznim podacima. Navedeni proces modelu daje širok raspon gledišta iz kojih može učiti, te mu na taj način omogućuje da brže

uočava uzorke. Ova je metoda posebno korisna kada je prikupljanje velikog i raznovrsnog skupa podataka teško ili dugotrajno. Modeli mogu doseći više razine performansi i pokazati vrhunsku prilagodljivost poboljšavanjem postojećih podataka putem proširenja.



Slika 17. Geometrijska transformacija



Slika 18. Transformacija boja



Slika 19. Kernel filter

Povećanje podataka u ovom se zadatku odvijalo kako je prikazao na slici [Slika 20.]. Lambda funkcija se koristi s funkcijom *map* za ponavljanje skupa podataka za obuku. Za svaki par slika-oznaka, lambda funkcija primjenjuje povećanje podataka na sliku, posebno postavljajući oznaku obuke na *True*. Ovo označava da će se povećanje primijeniti tijekom faze obuke. Proširena slika, zajedno sa svojom izvornom oznakom, zatim se oblikuje u tuple. Kao rezultat toga, izgrađen je skup podataka *augmented_train_ds* koji sada sadrži ove proširene parove slika-oznaka.

```
augmented_train_ds = train_ds.map(lambda x, y: (data_augmentation(x, training=True), y))
```

Slika 20. Linija koda za augmentaciju podataka

6.1.3. Obuka modela s proširenim podacima

Nakon povećanja podataka model prolazi kroz proces obuke kako je prikazano na slici [Slika 21.]. Epohe parametra postavljene su na 10 (*epochs = 10*) što znači da će toliko puta čitav skup podataka za obuku biti prosljeđen naprijed i natrag kroz neuronsku mrežu. *Resnet_model* obučava se pomoću metode *fit*. Podatci o obuci, predstavljeni ranije definiranom varijablom *augmented_train_ds*, koriste se kao ulaz, dok je skup podataka za provjeru valjanosti (*val_ds*) osiguran za praćenje izvedbe modela na još neviđenim podacima. Tijekom svake epohe model uči predviđati, procjenjuje gubitak i prilagođava svoje unutarnje parametre kako bi poboljšao svoju točnost. Ovaj se proces ponavlja za određeni broj epoha, postupno usavršavajući sposobnost modela da klasificira slike. Povijest treninga, koja sadrži važne metrike poput gubitka i točnosti tijekom svake epohe, pohranjuje se u varijablu *history* za kasniju analizu ili vizualizaciju.

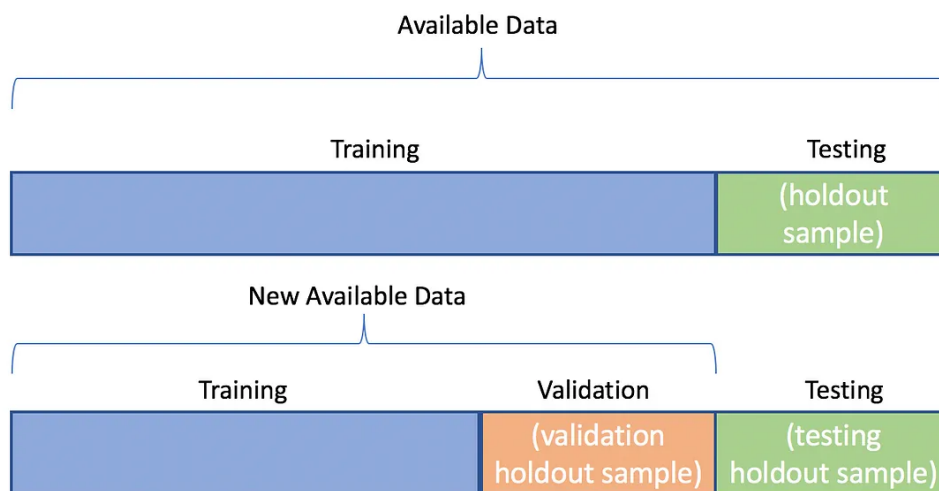
```
epochs = 10
history = resnet_model.fit(
    augmented_train_ds,
    validation_data=val_ds,
    epochs=epochs
)
```

Slika 21. Prikaz koda za obuku modela

7. Evaluacija modela

Razvoj učinkovitih i pouzdanih neuronskih mreža kao ključnu fazu uključuje evaluaciju modela. Ponekad se još i naziva skupom testnih podataka, a koristi se kao mjerilo za procjenu sposobnosti modela da napravi precizna predviđanja na prethodno neopaženim podacima. Ovim korakom možemo naučiti više o sposobnosti modela da generalizira svoje otkrivene obrasce na stvarne situacije.

Jedna od najvažnijih stvari, kako bi se model odnosno njegova izvedba ispravno procijenili, jest da model nije treniran na cijelom skupu podataka. [13] Ovaj korak bio je spomenut još u poglavlju 3.1. kada se skup podataka dijelio na skup podataka za obuku (80% podataka) i skup podataka za validaciju (20%). Ovisno o veličini skupa podataka s kojim se kreće u rješavanje određenog zadatka, postoji i treći podskup koji se može uvesti a to je skup podataka za testiranje [Slika 22.]. Radi li se s manjim skupovima zadataka, kao što je slučaj u ovome radu, ili kada su računalni resursi ograničeni, praksa je usredotočiti se na skupove za obuku i validaciju. Izvedba modela može se procijeniti na skupu za provjeru valjanosti, a pretpostavlja se da će se dobra izvedba na skupu za provjeru dobro generalizirati na nove, nevidljive podatke.



Slika 22. Prikaz podjele skupova podataka

7.1. Metrike klasifikacije

Predviđanje oznaka klasa na temelju ulaznih podataka sastavni je dio problema klasifikacije strojnog učenja. U binarnoj klasifikaciji, primjerice, rezultat spada u jednu od dvije kategorije, stvarajući na taj način jasnu dihotomiju. Ovakav pristup može se susresti u scenarijima kao što je otkrivanje neželjene pošte, gdje se poruke e-pošte dijele u kategorije "neželjena pošta" ili "nije neželjena pošta", a povremeno se za predstavljanje dviju klasa koriste i alternativni izrazi poput 'pozitivno' i 'negativno' ili 'klasa 1' i 'klasa 0'. [14]

Ocjenjivanje izvedbe modela binarne klasifikacije bitan je korak u osiguravanju njegove učinkovitosti te u tu svrhu postoji nekoliko dobro utvrđenih metrika. Među njima, i u širokoj primjeni, mogu se pronaći točnost (*Accuracy*), preciznost (*Precision*), prisjećanje (*Recall*), F1 score, AUC-ROC te matrica konfuzije. [14]

7.1.1. Točnost (Accuracy)

Točnost je najjednostavnija metrika za procjenu modela i mjeri koliko često klasifikator točno predviđa. Možemo ju definirati kao omjer broja točnih predviđanja i ukupnog broja predviđanja.

Ukoliko bi se za neki model dobila stopa točnosti od 99% moglo bi se pretpostaviti kako taj model ima vrlo dobre rezultate, ali to nije uvijek točno i može dovesti u zabludu u nekim situacijama.

Na primjer, imate li skup podataka s dvije ciljane klase koji sadrži 100 uzoraka gdje 98 uzoraka pripada klasi A, a 2 uzorka pripadaju klasi B. U podacima za obuku, model bi dao 98% točnosti te zato moramo pogledati više metrika kako bismo dobili bolji rezultat. [15]

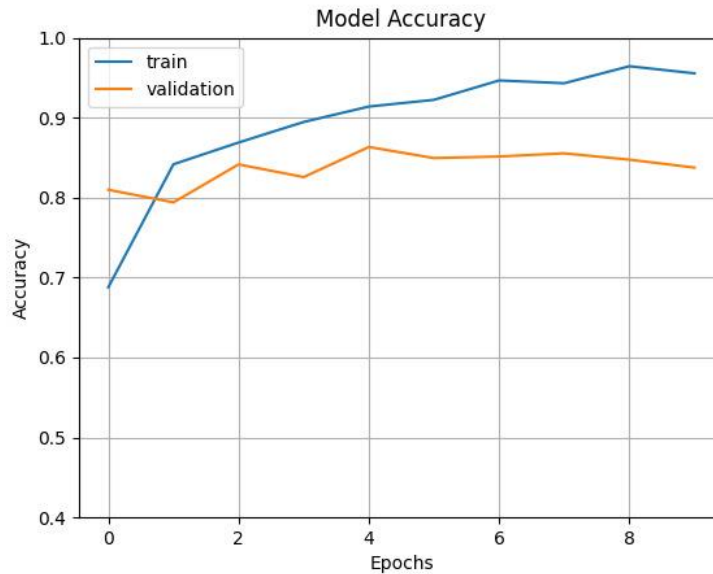
Ovom metodom evaluiran je i model za klasifikaciju otpada čija je obuka bila prikazana u prijašnjim poglavljima. Točnost modela za obuku i provjeru valjanosti prikazat će se u linijskom dijagramu kojega generira kod prikazan na slici [Slika 23.].

```
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0.4, ymax=1)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```

Slika 23. Dio koda za generiranje linijskog dijagrama koji prikazuje točnost

Naredba `fig1 = plt.gcf()` inicira stvaranje nove figure (slike), omogućujući daljnju prilagodbu. Nakon toga se izrazi `plt.plot(history.history['accuracy'])` i `plt.plot(history.history['val_accuracy'])` koriste za iscrtavanje točnosti obuke i provjere valjanosti koristeći podatke iz ranije definirane varijable `history`. Ograničenja y-osi postavljena su između 0,4 i 1 s `plt.axis(ymin=0,4, ymax=1)` kako bi se osigurao određeni raspon za bolju vizualizaciju.

Nakon pokretanja koda dijagram točnosti izgleda ovako [Slika 24.]:



Slika 24. Dijagram točnosti

Iz dijagrama točnosti može se iščitati kako se kroz zadanih 10 epoha točnost modela povećava s otprilike 0.7 na iznad 0.9 za skup podataka za uvježbavanje dok ostaje blizu 0.8 za skup podataka za validaciju. Model, kako je i očekivano u ranim fazama obuke, postaje sve bolji u uklapanju podataka o obuci što dokazuje njena sve veća točnost. Jednako je važno obratiti pozornost na točnost valjanosti. Njegova konstantna vrijednost od 0,8 sugerira da se model prilično učinkovito generalizira na nepoznate podatke. Ovo je dobar pokazatelj koji da model nije pretjerano prilagođen tj. da se obrasci stvarnog učenja mogu primijeniti na svježije podatke.

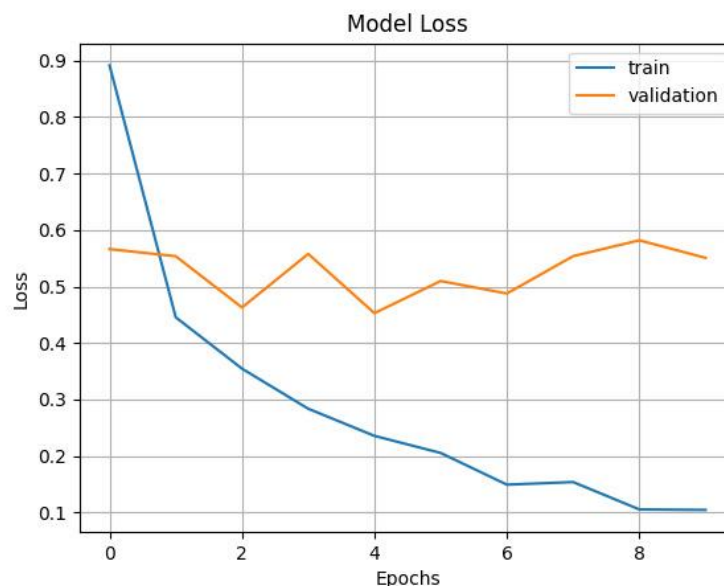
Oblikovanje uspješnog modela, osim točnosti, zahtjeva i odabir odgovarajuće funkcije gubitka (*loss*). Procjenom podudarnosti između predviđanja modela i stvarnih oznaka istine na terenu, ova ključna komponenta mjeri izvedbu modela tijekom treninga. Što su predviđanja modela bliža stvarnim vrijednostima, manja je vrijednost gubitka. Funkcija gubitka usmjerava proces optimizacije prema identificiranju najbolje moguće zbirke parametara za smanjenje ukupne pogreške. Kao rezultat toga, interakcija između točnosti i funkcije gubitka nudi temeljitu procjenu sposobnosti modela za razumijevanje i ekstrapolaciju podataka. Jednostavno rečeno, funkcija gubitka pokazuje koliko je model netočan u određivanju odnosa između x i y odnosno koliko je trenutni izlaz algoritma udaljen od željenog. [16]

Kako bi se, kao i dijagram točnosti, prikazao i dijagram koji ilustrira gubitak obuke i validacije tijekom epoha procesa obuke modela strojnog učenja, potrebno ga je generirati kodom prikazanim na slici [Slika 25.].

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.grid()
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```

Slika 25. Dio koda za generiranje linijskog dijagrama koji prikazuje gubitak

Dijagram, prikazan na slici [Slika 26.], generira se na manje-više isti način koji je objašnjen kod točnosti [Slika 25.] te ga nije potrebno dodatno komentirati.



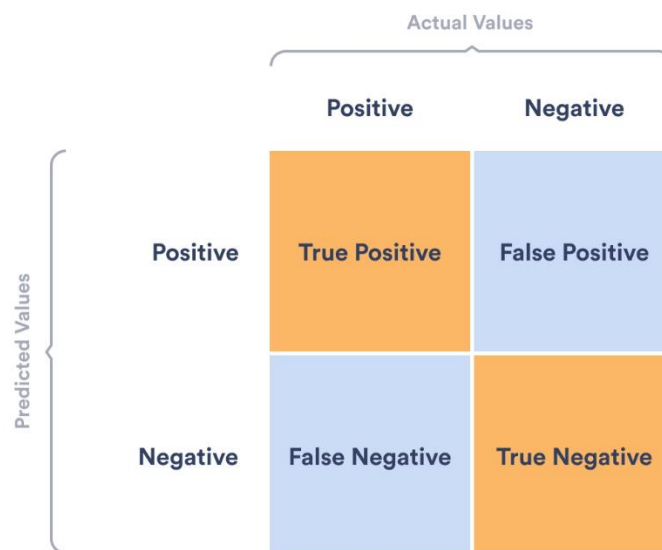
Slika 26. Dijagram funkcije gubitka

Dijagram pokazuje kako je gubitak kod uvježbavanja pao s 0.9 na 0.1 tijekom 10 epoha što ukazuje na to da model odlično obavlja posao prilagođavanja skupa podataka uvježbavanja. To je znak da model uspješno uči iz treninga, što je ohrabrujuće. Gubitak validacije, međutim, kreće se od 0,5 do 0,6. U usporedbi s gubitkom treninga, to je ipak malo više što implicira da bi model mogao biti pomalo pretreniran odnosno model ima izvanredne rezultate na podacima za obuku, ali se muči na neisprobanim podacima

7.1.2. Matrica konfuzije

Matrica konfuzije ili matrica pogrešaka također je jedan od alata u području strojnog učenja. Prikazuje broj točnih i netočnih predviđanja napravljenih modelom u usporedbi sa stvarnim klasifikacijama u testnom skupu ili vrstom pogrešaka koje su napravljene. Ova matrica opisuje izvedbu modela klasifikacije na testnim podacima za koje su poznate prave vrijednosti. To je $n \times n$ matrica, gdje je n broj klasa. Ova se matrica može generirati nakon predviđanja na temelju testnih podataka. [15]

Matricu konfuzije, prikazanu na slici [Slika 27.] najlakše je objasniti na primjeru pa se tako može zamisliti se radi o modelu koji uči razlikovati fotografije mačke od pasa. Postoje četiri moguća rezultata iz predviđanja koje daje. Kada model ispravno identificira psa kao psa, to je poznato kao pravi pozitivan (TP). Onda kada ispravno predvidi mačku kao mačku, proizvodi istinite negative (TN). Lažno pozitivni rezultati (FP), s druge strane, javljaju se kada sustav predvidi psa dok je na fotografiji zapravo mačka, a posljednje, ali ne i manje važni, lažno negativni rezultati (FN) događaju se kada model predviđa mačku umjesto, ispravno, psa. Matrica zabune daje jasnu sliku performansi modela urednim grupiranjem ovih rezultata.



Slika 27. Matrica konfuzije

Isječak koda koji je odgovoran za generiranje matrice konfuzije prikazan je na slici [Slika 28.]. Cilj procijeniti koliko je dobro uvježbani ResNet-50 model radio na validacijskom skupu podataka Najprije se korištenjem naredbe `resnet_model.predict(val_ds)` izrađuju predviđanja za fotografije iz skupa podataka za validaciju. Oznake predviđene (`predicted_classes`) klase zatim se izdvajaju identificiranjem indeksa klase s najvećom predviđenom vjerojatnošću za svaku sliku. Istovremeno, stvarne oznake klase (`true_classes`) prikupljaju se iz validacijskog skupa podataka. Matrica zabune stvara se pomoću ovih vrijednosti nakon što oba skupa oznaka postanu dostupna. Broj pravih pozitivnih, pravih negativnih, lažno pozitivnih i lažno negativnih rezultata prikazan je u matrici zabune, koja daje temeljitu procjenu

izvedbe modela. Matrica konfuzije se predstavlja kao toplinska karta, pri čemu svaka ćelija predstavlja određenu mješavinu pravih i očekivanih klasa. Numeričke vrijednosti ćelija daju točan broj. Y-os predstavlja stvarne razrede, a x-os predstavlja očekivane razrede. Ovaj vizualni prikaz nudi pronicljive informacije o prednostima i potencijalnim slabostima modela u zadacima kategorizacije.

```

predictions = resnet_model.predict(val_ds)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = np.concatenate([y for x, y in val_ds], axis=0)

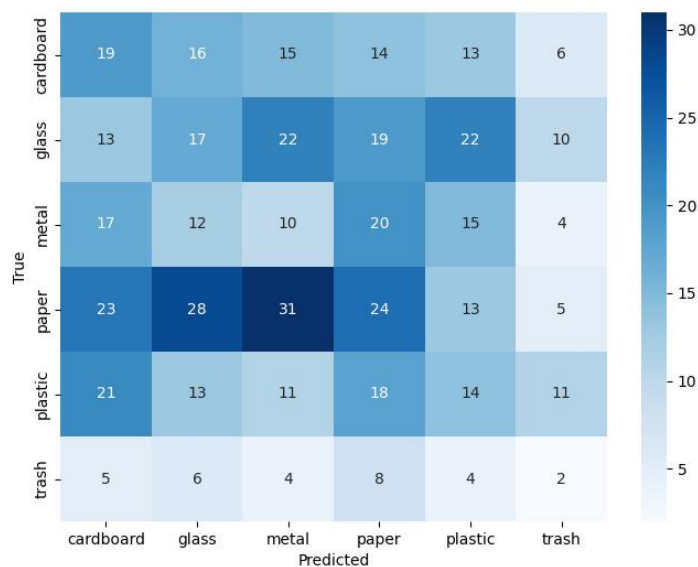
confusion_mtx = confusion_matrix(true_classes, predicted_classes)

plt.figure(figsize=(8,6))
sns.heatmap(confusion_mtx, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

```

Slika 28. Dio koda koji prikazuje stvaranje matrice konfuzije

Ovim kodom generirala se matrica konfuzije koja se može vidjeti na slici [Slika 29.]. Iz slike matrice konfuzije vidljivo je da su predviđanja modela raspršena u nekoliko klasa, u skladu s disperziranom matricom zabune, koja pokazuje da ne postoji čvrsta linija koja razdvaja različite skupine.



Slika 29. Konačna matrica konfuzije

8. Zaključak

Proces obuke modela strojnog učenja ključan je korak u izradi preciznih i pouzdanih predviđanja. Moguće je naučiti mnogo o izvedbi modela analizom njegove točnosti i gubitka tijekom niza epoha. Model je uspješno naučio temeljne obrasce u podacima kada postoji stalno poboljšanje točnosti uvježbavanja i odgovarajući pad u gubitku uvježbavanja. Kako bismo bili sigurni da se model učinkovito generalizira na prethodno neprijavljene podatke, bitno je procijeniti izvedbu modela na drugom skupu za provjeru valjanosti. Neznatno povećavanje točnosti provjere valjanosti ili njeno zaustavljanje dok točnost uvježbavanja primjetno raste može biti znak pretjeranog treniranja. Pretjerano treniranje nedostatak je generalizacije koji proizlazi iz toga što model postaje previše specijaliziran za prikupljanje šuma ili određenih informacija u podacima o obuci. Za rješavanje ovog problema mogu se koristiti tehnike poput regularizacije, ispadanja ili smanjenja složenosti modela.

Raspršena matrica konfuzije, kakva je generirana u poglavlju 7.1.2. na slici [Slika 29.] ukazuje na to da model ima poteškoća u razlikovanju nekoliko klasa. Složenost modela, nedostatak podataka o obuci, neuravnotežena klasa, loši hiperparametri ili inženjering značajki nekoliko su mogućih uzroka.

U slučaju skupa podataka korištenih u svrhu ovoga zadatka problem zasigurno nije u neuravnoteženim klasama budući da su one podjednake. Uzrok problema bi tako mogao biti loše postavljene hiperparametri ili kvaliteta podataka. Naime, većina fotografija otpada korištena za trening slikana je na jednakoj bijeloj površini te bi možda bilo potrebno uključiti različitije skupove, ali i povećati njihov broj. Također, možda bi bilo potrebno promijeniti arhitekturu modela ili možda uzeti u obzir funkcije gubitka ponderirane prema klasi kako bi se riješio navedeni problem.

Zaključno, faza obuke ključna je za stvaranje čvrstog modela strojnog učenja. Kako bi se poboljšao model i poboljšala njegova predviđanja potrebno je redovito pratiti pokazatelje izvedbe kao što su točnost i gubitak. Sposobnost modela da proizvede precizna predviđanja u aplikacijama u stvarnom svijetu može se poboljšati implementacijom usmjerenih promjena i razumijevanjem suptilnosti ponašanja modela tijekom obuke.

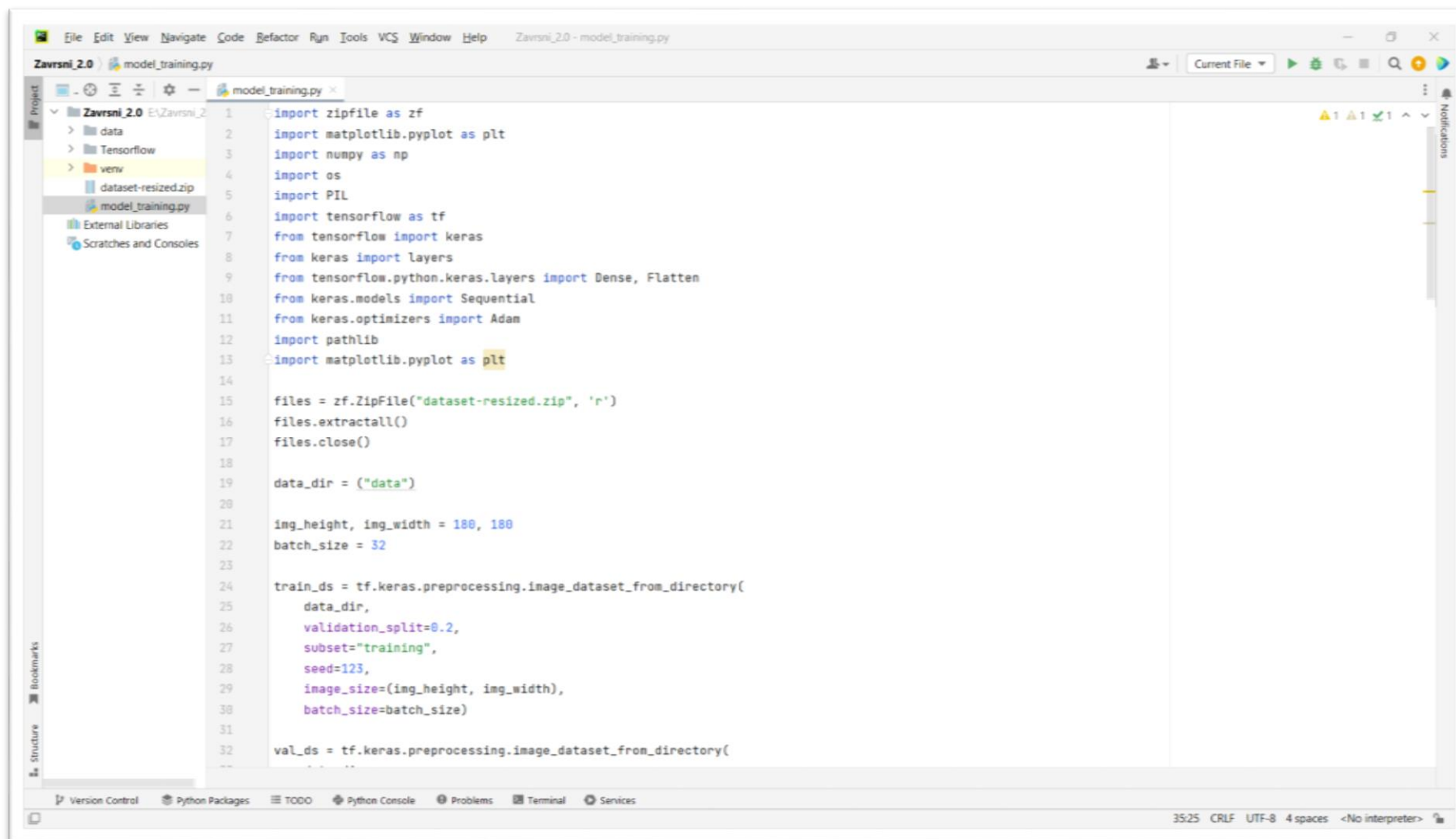
Reference

- [1]: Minsky, M. L., *Computation: Finite and Infinite Machines*. Prentice-Hall (1967), pristupljeno 16.9.2023.
- [2] What is deep learning? (<https://www.ibm.com/topics/deep-learning>), pristupljeno 25.8.2023.
- [3] Foster, D. *Generative Deep Learning* (2019.), pristupljeno 25.8.2023.
- [4] Introduction to Recurrent Neural Network (<https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>), pristupljeno 1.9.2023.
- [5]: Zachary C. Lipton, John Berkowitz, Charles Elkan, *A Critical Review of Recurrent Neural Networks for Sequence Learning* (2015), pristupljeno 1.9.2023.
- [6]: RECURRENT NEURAL NETWORK (RNN) (<https://medium.com/mllearning-ai/recurrent-neural-network-rnn-20a619190586>), pristupljeno 2.9.2023.
- [7]: Convolutional Neural Networks, Explained (<https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>), pristupljeno 2.9.2023.
- [8]: VGG Very Deep Convolutional Networks (VGGNet) (<https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/>), pristupljeno 2.9.2023.
- [9]: Deep Residual Networks (ResNet, ResNet50) – 2023 Guide (<https://viso.ai/deep-learning/resnet-residual-neural-network/>), pristupljeno 6.9.2023.
- [10]: The Vanishing Gradient Problem (<https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484>), pristupljeno 11.9.2023.
- [11]: *The State of Data Science 2020*, pristupljeno 26.8.2023.
- [12]: (Zip datoteka), <https://github.com/garythung/trashnet/blob/master/data/dataset-resized.zip>, pristupljeno 20.8.2023.
- [13]: Evaluating a machine learning model. (<https://www.jeremyjordan.me/evaluating-a-machine-learning-model/>), pristupljeno 11.9.2023.
- [14]: Home Metrics to Evaluate your Classification Model to take the right decisions (<https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/>), pristupljeno 12.9.2023.
- [15]: Evaluation Metrics for Classification Models (<https://medium.com/analytics-vidhya/evaluation-metrics-for-classification-models-e2f0d8009d69>), pristupljeno 15.9.2023.
- [16]: Basic Introduction to Loss Functions (<https://www.analyticsvidhya.com/blog/2022/08/basic-introduction-to-loss-functions/>), pristupljeno 15.9.2023.

Prilozi

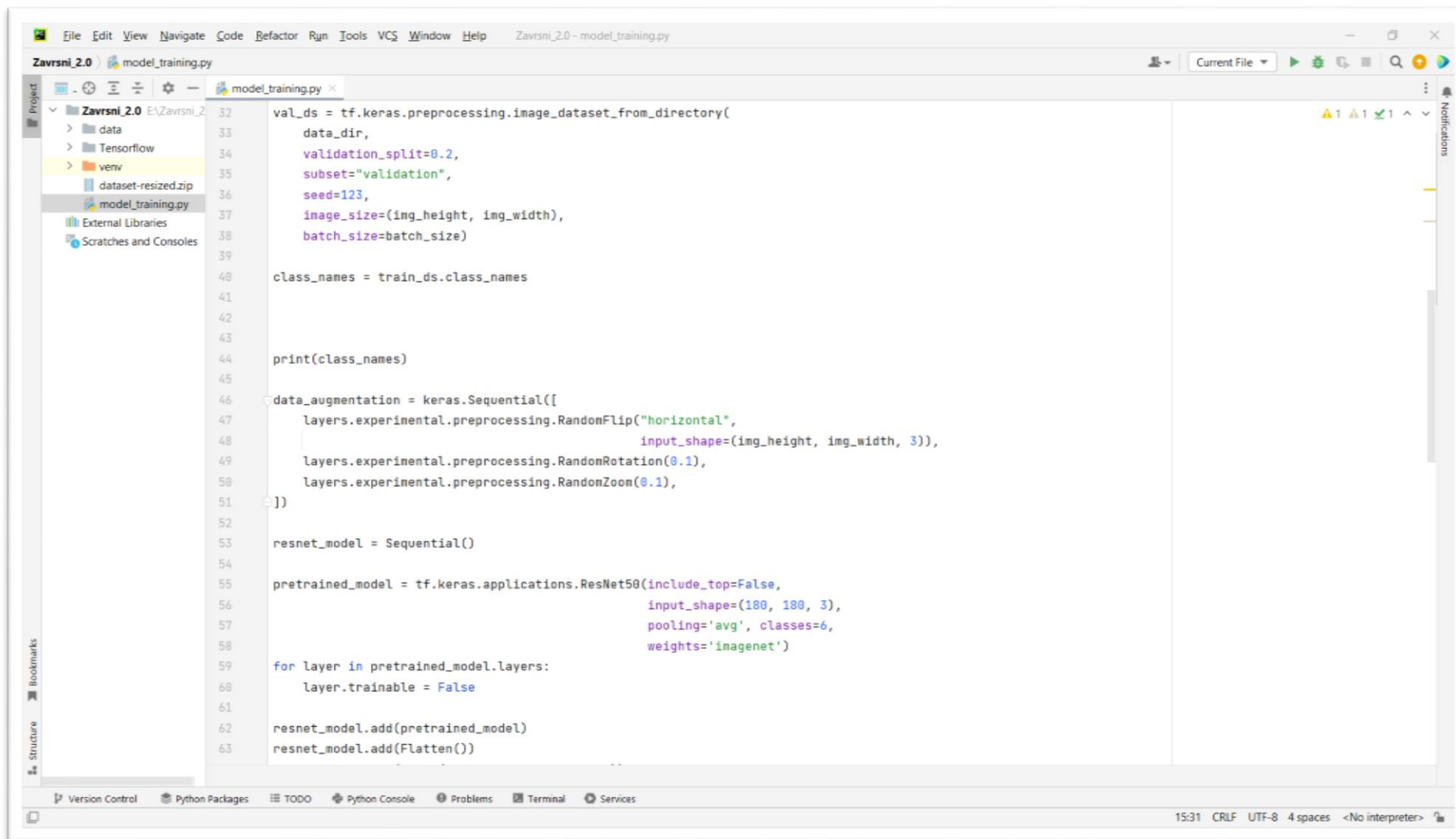
I: Programski kod

Prilog I: Programski kod



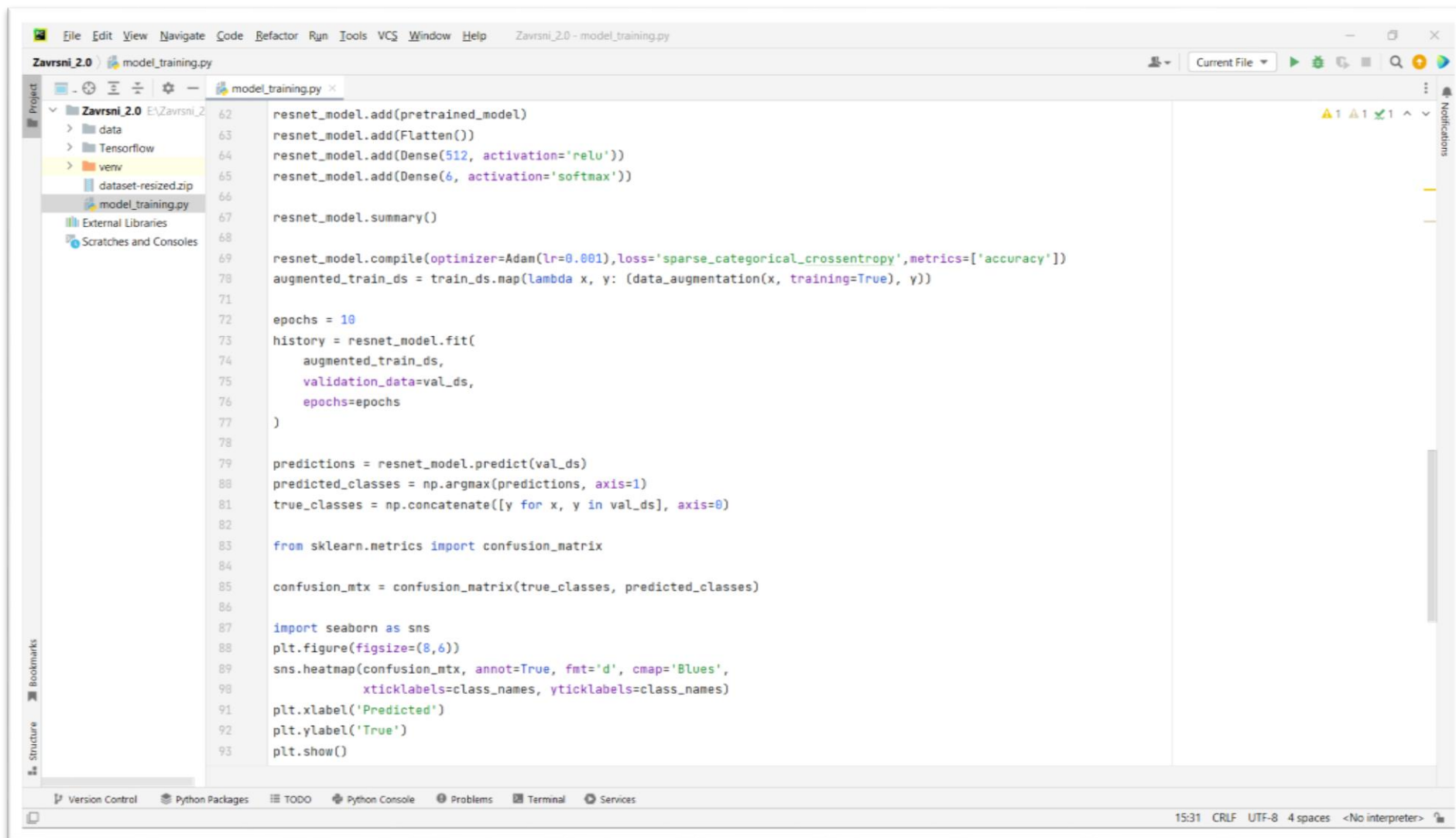
The image shows a screenshot of an IDE (likely PyCharm) with a Python file named `model_training.py` open. The code is as follows:

```
1 import zipfile as zf
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import os
5 import PIL
6 import tensorflow as tf
7 from tensorflow import keras
8 from keras import layers
9 from tensorflow.python.keras.layers import Dense, Flatten
10 from keras.models import Sequential
11 from keras.optimizers import Adam
12 import pathlib
13 import matplotlib.pyplot as plt
14
15 files = zf.ZipFile("dataset-resized.zip", 'r')
16 files.extractall()
17 files.close()
18
19 data_dir = ("data")
20
21 img_height, img_width = 180, 180
22 batch_size = 32
23
24 train_ds = tf.keras.preprocessing.image_dataset_from_directory(
25     data_dir,
26     validation_split=0.2,
27     subset="training",
28     seed=123,
29     image_size=(img_height, img_width),
30     batch_size=batch_size)
31
32 val_ds = tf.keras.preprocessing.image_dataset_from_directory(
```

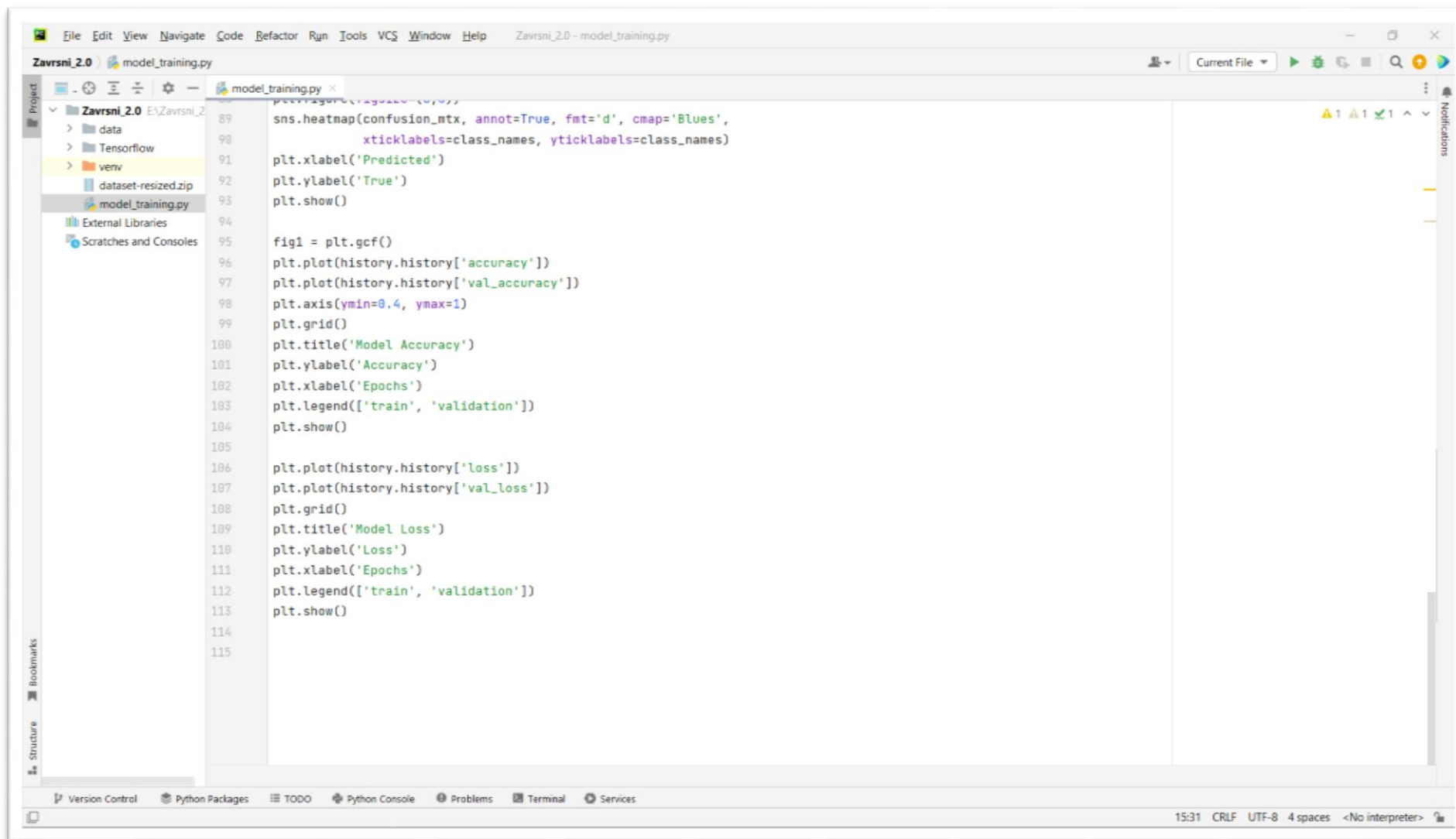



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Završni_2.0 - model_training.py
Završni_2.0 model_training.py
Project
  Završni_2.0
    data
    Tensorflow
    venv
    dataset-resized.zip
    model_training.py
  External Libraries
  Scratches and Consoles
Structure
Bookmarks
32 val_ds = tf.keras.preprocessing.image_dataset_from_directory(
33     data_dir,
34     validation_split=0.2,
35     subset="validation",
36     seed=123,
37     image_size=(img_height, img_width),
38     batch_size=batch_size)
39
40 class_names = train_ds.class_names
41
42
43
44 print(class_names)
45
46 data_augmentation = keras.Sequential([
47     layers.experimental.preprocessing.RandomFlip("horizontal",
48         input_shape=(img_height, img_width, 3)),
49     layers.experimental.preprocessing.RandomRotation(0.1),
50     layers.experimental.preprocessing.RandomZoom(0.1),
51 ])
52
53 resnet_model = Sequential()
54
55 pretrained_model = tf.keras.applications.ResNet50(include_top=False,
56     input_shape=(180, 180, 3),
57     pooling='avg', classes=6,
58     weights='imagenet')
59
60 for layer in pretrained_model.layers:
61     layer.trainable = False
62
63 resnet_model.add(pretrained_model)
64 resnet_model.add(Flatten())
```

15:31 CRLF UTF-8 4 spaces <No interpreter>



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Završni_2.0 - model_training.py
Završni_2.0 model_training.py
Project
  Završni_2.0
    data
    Tensorflow
    venv
    dataset-resized.zip
    model_training.py
    External Libraries
    Scratches and Consoles
Structure
  model_training.py
62 resnet_model.add(pretrained_model)
63 resnet_model.add(Flatten())
64 resnet_model.add(Dense(512, activation='relu'))
65 resnet_model.add(Dense(6, activation='softmax'))
66
67 resnet_model.summary()
68
69 resnet_model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
70 augmented_train_ds = train_ds.map(lambda x, y: (data_augmentation(x, training=True), y))
71
72 epochs = 10
73 history = resnet_model.fit(
74     augmented_train_ds,
75     validation_data=val_ds,
76     epochs=epochs
77 )
78
79 predictions = resnet_model.predict(val_ds)
80 predicted_classes = np.argmax(predictions, axis=1)
81 true_classes = np.concatenate([y for x, y in val_ds], axis=0)
82
83 from sklearn.metrics import confusion_matrix
84
85 confusion_mtx = confusion_matrix(true_classes, predicted_classes)
86
87 import seaborn as sns
88 plt.figure(figsize=(8,6))
89 sns.heatmap(confusion_mtx, annot=True, fmt='d', cmap='Blues',
90             xticklabels=class_names, yticklabels=class_names)
91 plt.xlabel('Predicted')
92 plt.ylabel('True')
93 plt.show()
Version Control Python Packages TODO Python Console Problems Terminal Services
15:31 CRLF UTF-8 4 spaces <No interpreter>
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Zavrnsni_2.0 - model_training.py
Zavrnsni_2.0 model_training.py
Project
  Zavrnsni_2.0
    data
    Tensorflow
    venv
    dataset-resized.zip
    model_training.py
  External Libraries
  Scratches and Consoles
89 plt.figure(figsize=(10, 10))
90 sns.heatmap(confusion_mtx, annot=True, fmt='d', cmap='Blues',
91             xticklabels=class_names, yticklabels=class_names)
92 plt.xlabel('Predicted')
93 plt.ylabel('True')
94 plt.show()
95
96 fig1 = plt.gcf()
97 plt.plot(history.history['accuracy'])
98 plt.plot(history.history['val_accuracy'])
99 plt.axis(ymin=0.4, ymax=1)
100 plt.grid()
101 plt.title('Model Accuracy')
102 plt.ylabel('Accuracy')
103 plt.xlabel('Epochs')
104 plt.legend(['train', 'validation'])
105 plt.show()
106
107 plt.plot(history.history['loss'])
108 plt.plot(history.history['val_loss'])
109 plt.grid()
110 plt.title('Model Loss')
111 plt.ylabel('Loss')
112 plt.xlabel('Epochs')
113 plt.legend(['train', 'validation'])
114 plt.show()
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671

```