Računalni model strojnog učenja za sortiranje otpada

Baronica, Dora

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 diplomske ispite
Povjerenstvo za završne i diplomske 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:	Datum predaje rada:	Predviđeni datumi obrane:
30. 11. 2022.	**1. rok:** 20. 2. 2023. **2. rok (izvanredni):** 10. 7. 2023. **3. rok:** 18. 9. 2023.	**1. rok:** 27. 2. – 3. 3. 2023. **2. rok (izvanredni):** 14. 7. 2023. **3. rok:** 25. 9. – 29. 9. 2023.

Zadatak zadao: Doc. dr. sc. Tomislav Stipančić

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

Popis slika

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

Eksponencijalna 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 zaštiti 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 revolucionrana 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, ubraznim se korako razvila u veliko područje niza metodologija koje uključuju, primjerice, strojno učenje i obradu prirodnog jezika.

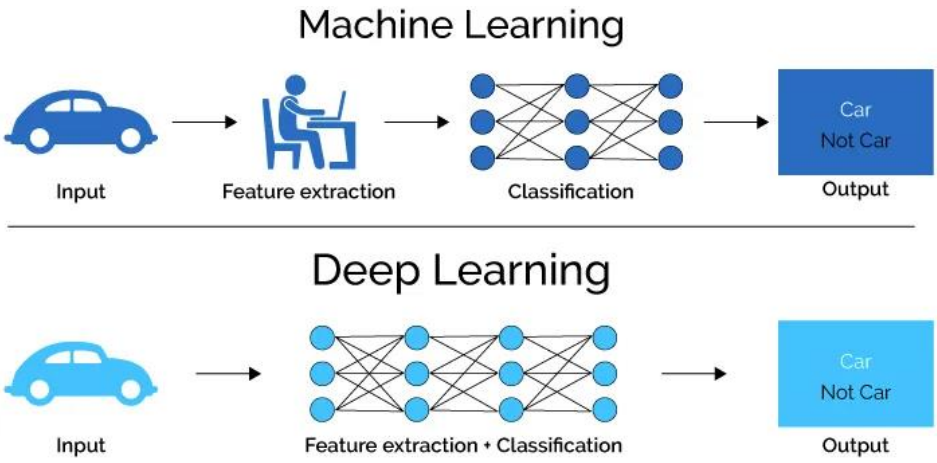
Evolucijska putanja umjetne inteligencije prati zadivljujuće putovanje od rudimentarnih simboličkih sustava razmišljanja do današnjeg doba kojim dominiraju strojno učenje i zamršene neuronske mreže. Kako navedene tehnologije napreduju, pomiču granice onoga za što su strojevi sposobni, donoseći dramatične 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.



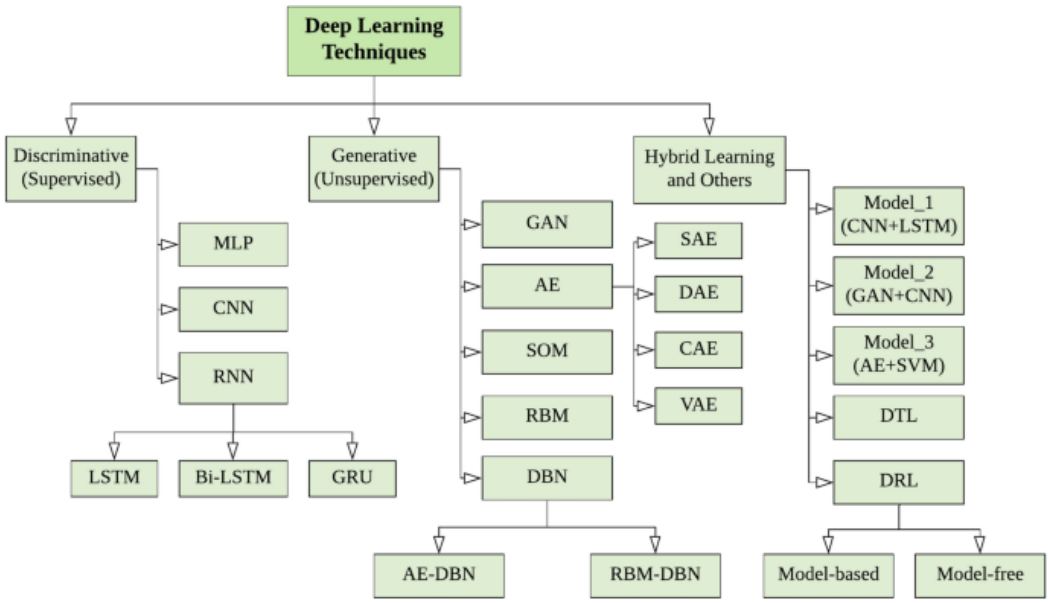
Slika 1. Razlika strojnog učenja i dubokog učenja

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 podataka 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.

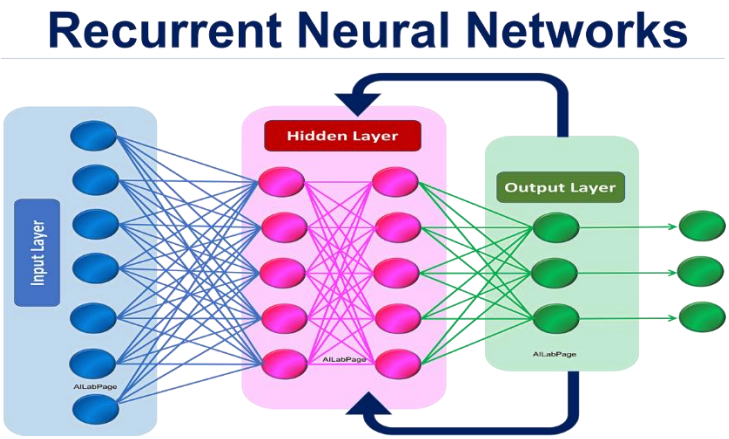
Drugim riječima, diskriminativno modeliranje pokušava procijeniti vjerojatnost da opažanje x pripada kategoriji y. Generativno modeliranje ne mari za označavanje opažanja. Umjesto toga, pokušava procijeniti vjerojatnost da se opažanje uopće vidi. [3]

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

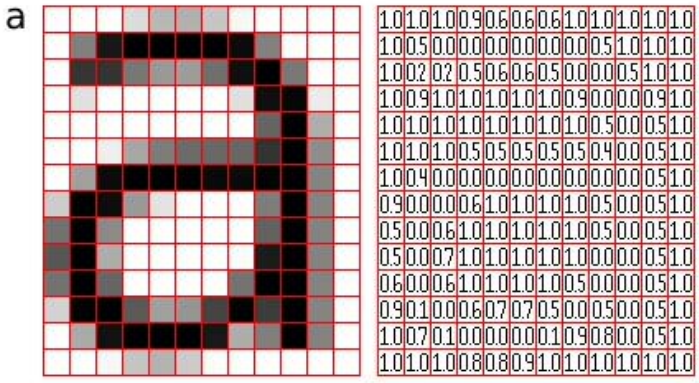
Rekurentne neuronske mreže vrlo su moćan alat za rad te se široko primjenjuju u svakodnevnom životu. Zbog napretka u arhitekturi modela, algoritama za obuku i paralelnog računanja tijekom proteklih trideset godina, prestale su biti modeli

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

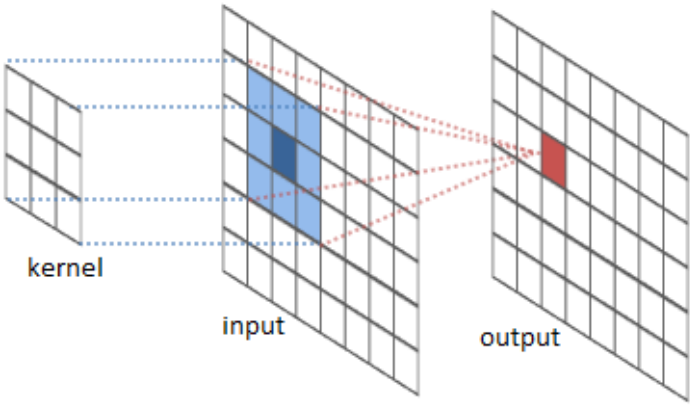
Jedinstvena vrsta algoritama koji, u golemom području umjetne inteligencije, imaju nevjerojatnu sposobnost da percipiraju svijet poput ljudi nazivaju se konvolucijske neuronske mreže (CNN). Može ih se zamisliti kao digitalne umjetnike koji vole uzorke, rubove i oblike.
Konvolucijske neuronske mreže pažljivo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

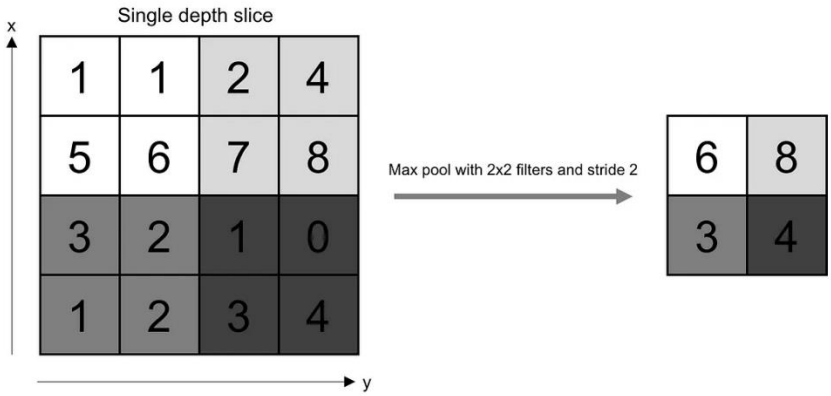
Slojevi konvolucijske neuronske mreže raspoređeni su tako da najprije otkrivaju jednostavnije uzorke poput linija i krivulja, gotovo kao da se slika skicira od nule. Zatim u sljedećem koraku prelaze na složenije detalje kao što su lica i objekti. Prepoznaju oblike unutar oblika, teksture unutar tekstura,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

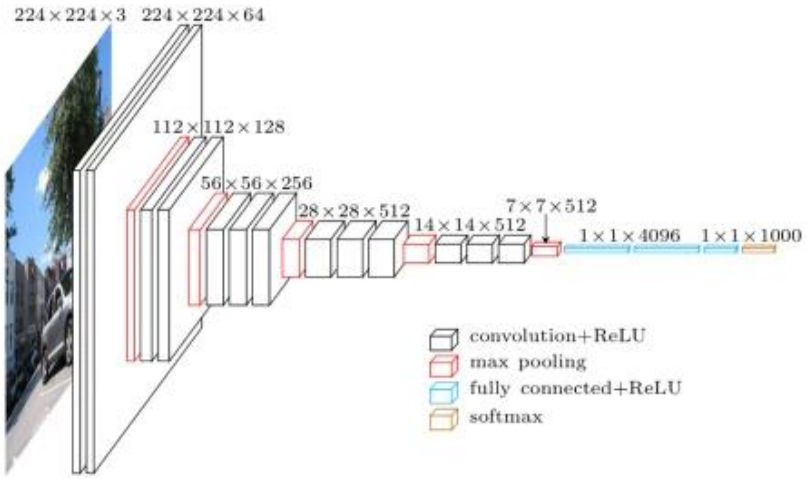
Cjelokupna struktura i raspored navedenih slojeva, uključujući njihovu vrstu i broj te njihove veze i protok informacija između njih, definiraju arhitekturu neuronske mreže. Izbor arhitekture neuronske mreže značajno utječe na njenu sposobnost da uči iz podataka. Dobro dizajnirana arhitektura može poboljšati 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 popilarnih CNN arhitektura 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 arhitektura 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" .

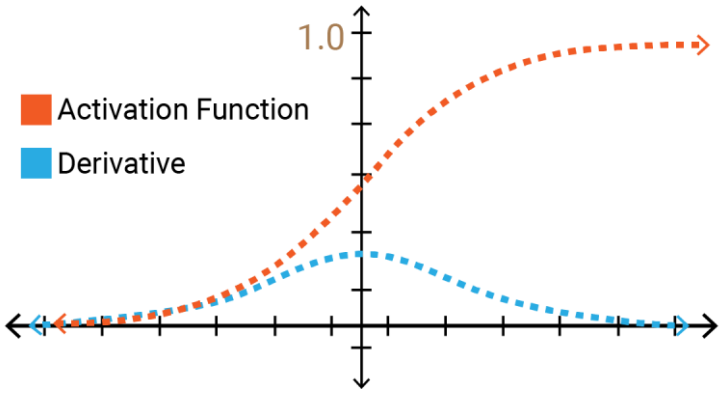
Sve u svemu, mreže VGG pokazale su izvanrednu učinkovitost u izvlačenju složenih informacija iz fotografija unatoč visokim zahtjevima za obradu. Iako su se pojavile novije arhitekture, VGG je i dalje kamen temeljac i standard u stvaranju konvolucijskih neuronskih mreža (CNN). Utjecaj u primjeni naglašava 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 podatcima. One ipak mogu stvarati probleme kod dubokih mreža kako je prikazano na slici [Slika 8.]. na primjeru Sigmoid aktivacijske funkcije.



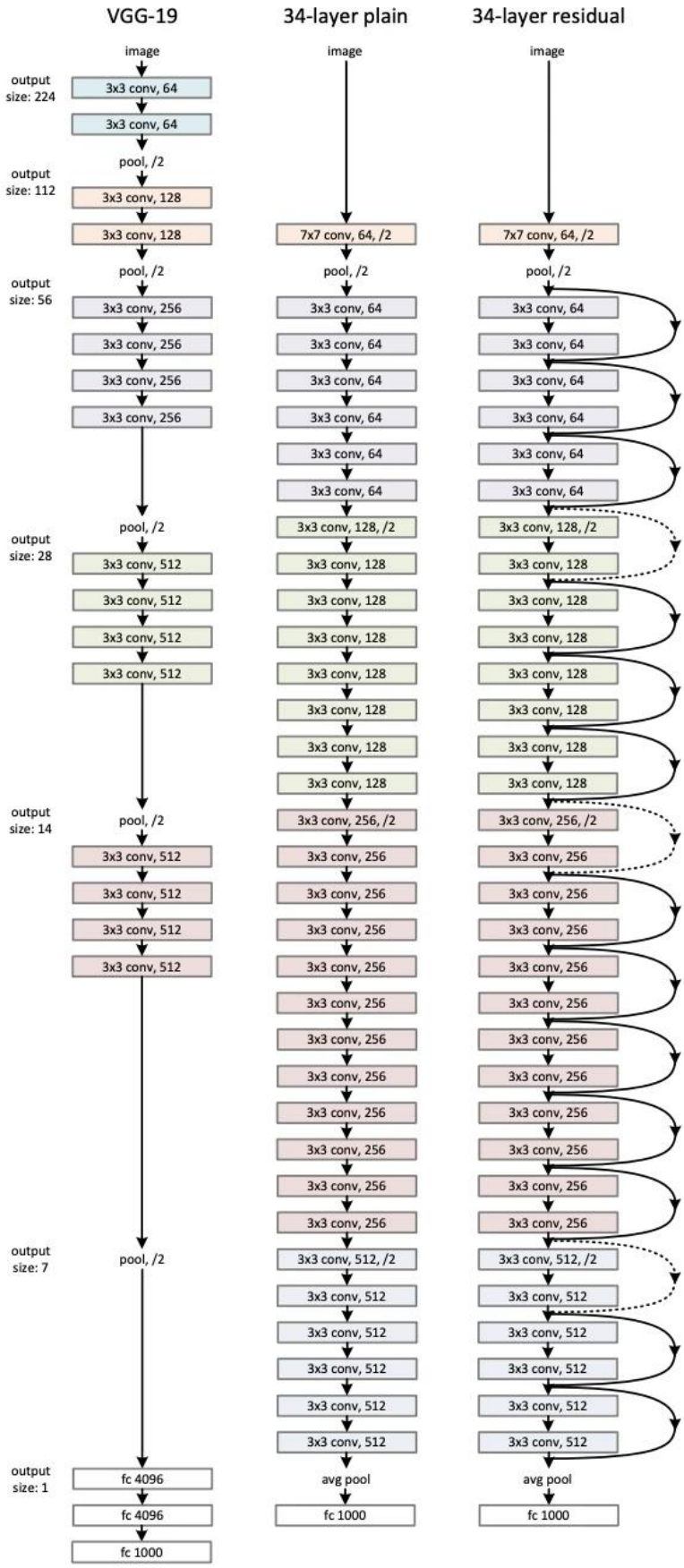
Slika 8. Problem nestajanja gradijenta (Sigmoid funkcija)

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 zamislit 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 poslovica „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]

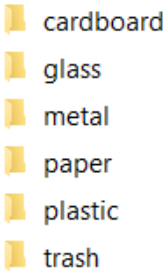
Potrebe problematike ovog rada zadovoljit će skup podataka u obliku fotografija raznih vrsta otpada smještenih u zip datoteku. [12]

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 ekstradiranja podataka, direktorij *data*, u kojemu su podatci pohranjeni, spremit će se u varijablu *data_dir*, ako je prikazano na slici [Sllika 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 pipline) 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 podatcima.

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.

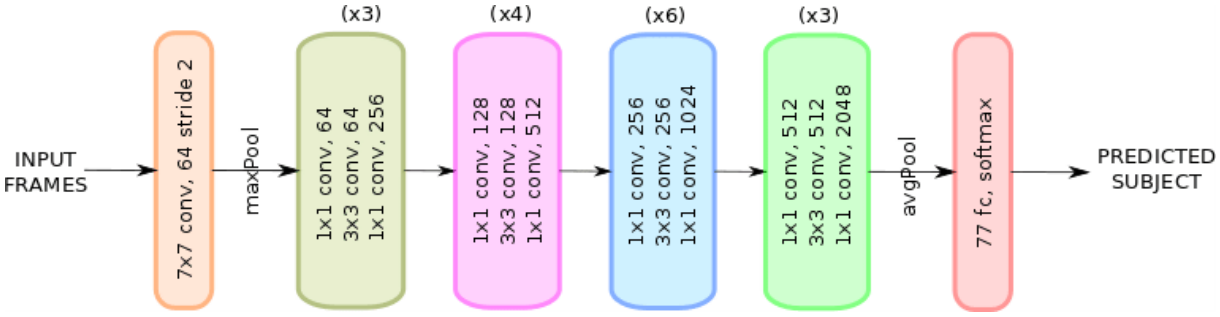
Model otkriva temeljne korelacije i obrasce u podacima o obučavanju, što mu omogućuje precizna predviđanja na prethodno neopaženim uzorcima.

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 skolpu 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 podatcima.

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 obučiti 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.

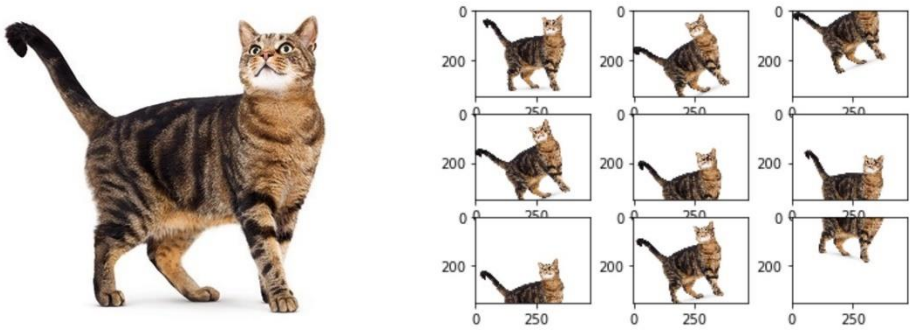
Stopa ili brzina učenja (*learning_rate*), koja u ovom slučaju iznosi 0,001, je hipermarametar koji određuje veličinu koraka pri kojoj se parametri modela ažuriraju tijekom obuke. Što je stopa učenja veća dolazi do brže kovergencije, ali ona isto tako može dovesti do rizika prekoračenja optimalnih vrijednosti. 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.

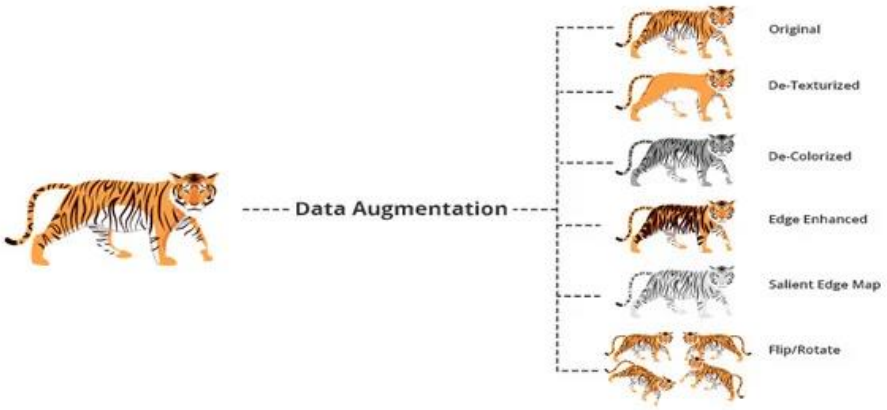
6.1.2. Augmentacija podataka

Područja računalnog vida i strojnog učenja uvelike se oslanjaju na koncept augmentacije odnosno povećanja podataka. Ono podrazumijeva izvođenje transformacija nad postojećim podatcima, 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 odvilo 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 podatcima

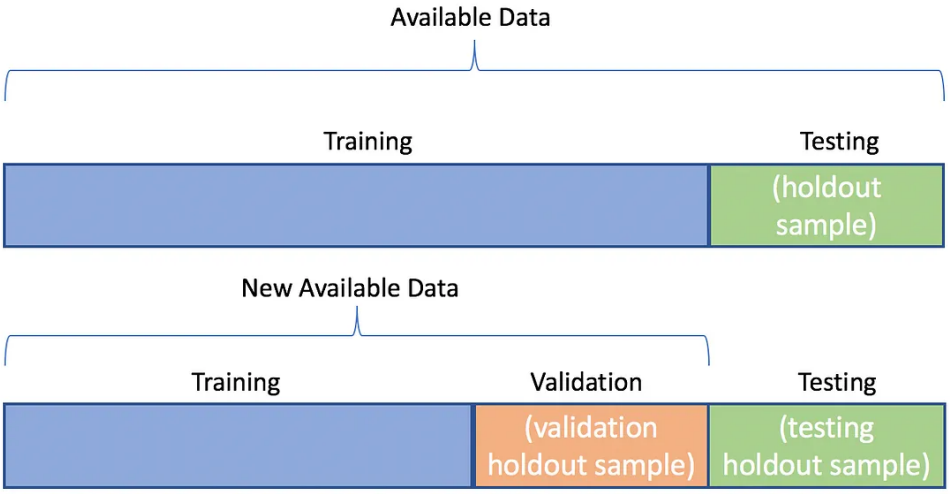
Nakon povećanja podataka model prolazi kroz proces obuke kako je prikazano na slici [Slika 21.]. Epohe parametra postavljene su na 10 (*epochs = 10*) što znači da će toliko puta čitav skup podataka za obuku biti proslijeđen naprijed i natrag kroz neuronsku mrežu. *Resnet_model* obučava se pomoću metode *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 podatcima. 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]

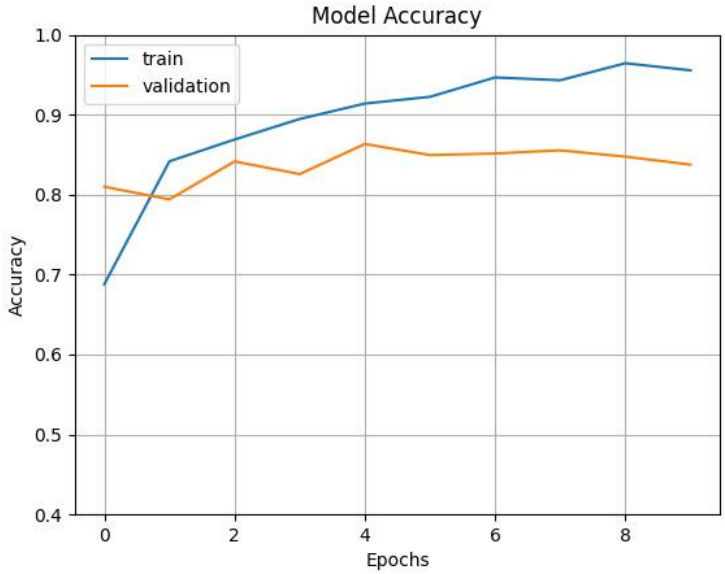
7.1.1. Točnost (Accuracy)

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 ciljne klase koji sadrži 100 uzoraka gdje 98 uzoraka pripada klasi A, a 2 uzorka pripadaju klasi B. U podatcima 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 svježe podatke.

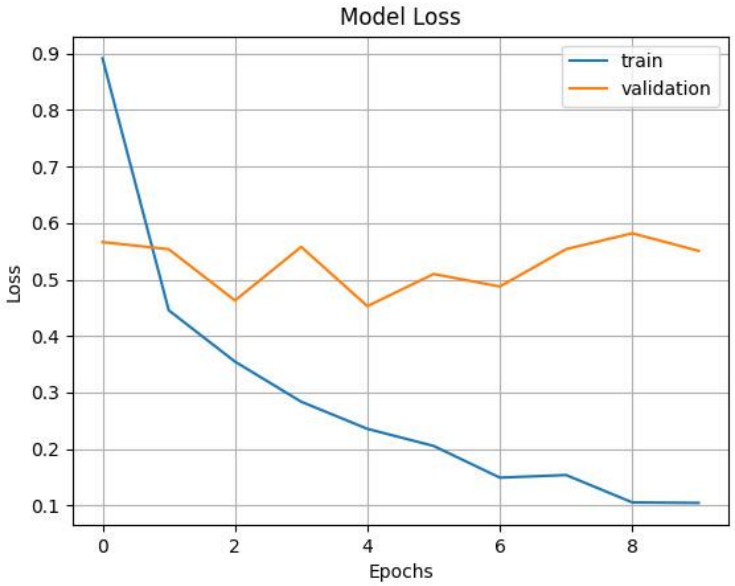
Oblikovanje uspješnog modela, osim točnosti, zahtjeva i odabir odgovarajuće funkcije gubitka (*loss*). Procjenom podudarnosti između predviđanja modela i stvarnih oznaka istine na terenu, ova ključna komponenta mjeri izvedbu modela tijekom treninga. Što su predviđanja modela bliža stvarnim vrijednostima, 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.

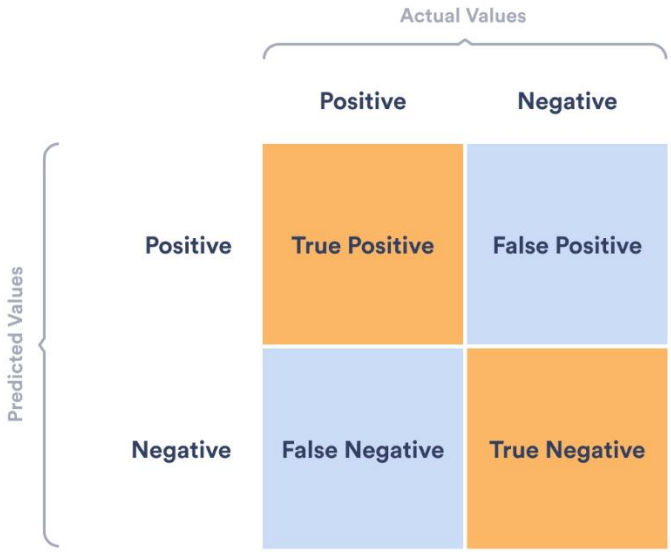


Slika 26. Dijagram funkcije gubitka

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 podatcima

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*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

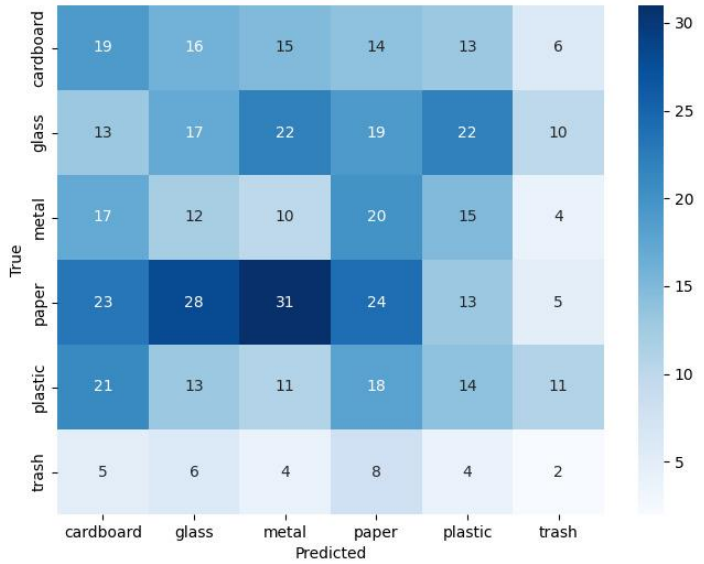
Isječak koda koji je odgovoran za generiranje matrice konfuzije prikazan je na slici [Slika 28.]. Cilj procijeniti koliko je dobro uvježbani ResNet-50 model radio na validacijskom skupu podataka Najprije se korištenjem naredbe *resnet_model.predict(val_ds*) izrađuju predviđanja za fotografije iz skupa podataka za validaciju. Oznake predviđene (*predicted_classes*) klase zatim se izdvajaju identificiranjem indeksa klase s najvećom predviđenom vjerojatnošću za svaku sliku. Istovremeno, stvarne oznake klasa (*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 podatcima 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 postavljeni hipermarametri 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), prisupljeno 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/mlearning-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/), pristupljebo 12.9.2023.

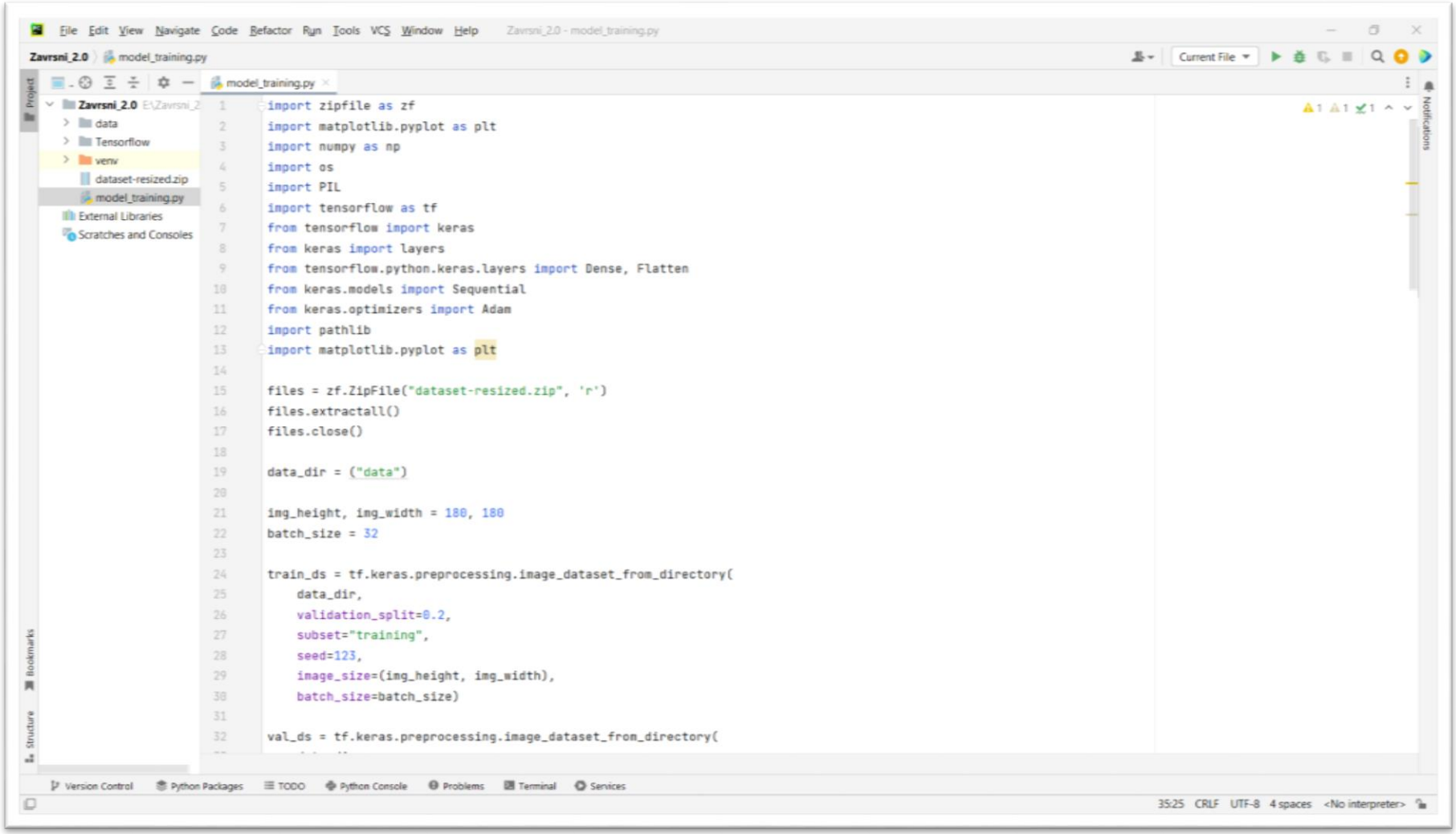
[15]: Evaluation Metrics for Classification Models (https://medium.com/analytics-vidhya/evaluation-metrics-for-classification-models-e2f0d8009d69), pristupljeno 15.9.2023.

[16]: Basic Introduction to Loss Functions (https://www.analyticsvidhya.com/blog/2022/08/basic-introduction-to-loss-functions/), pristupljeno 15.9.2023.

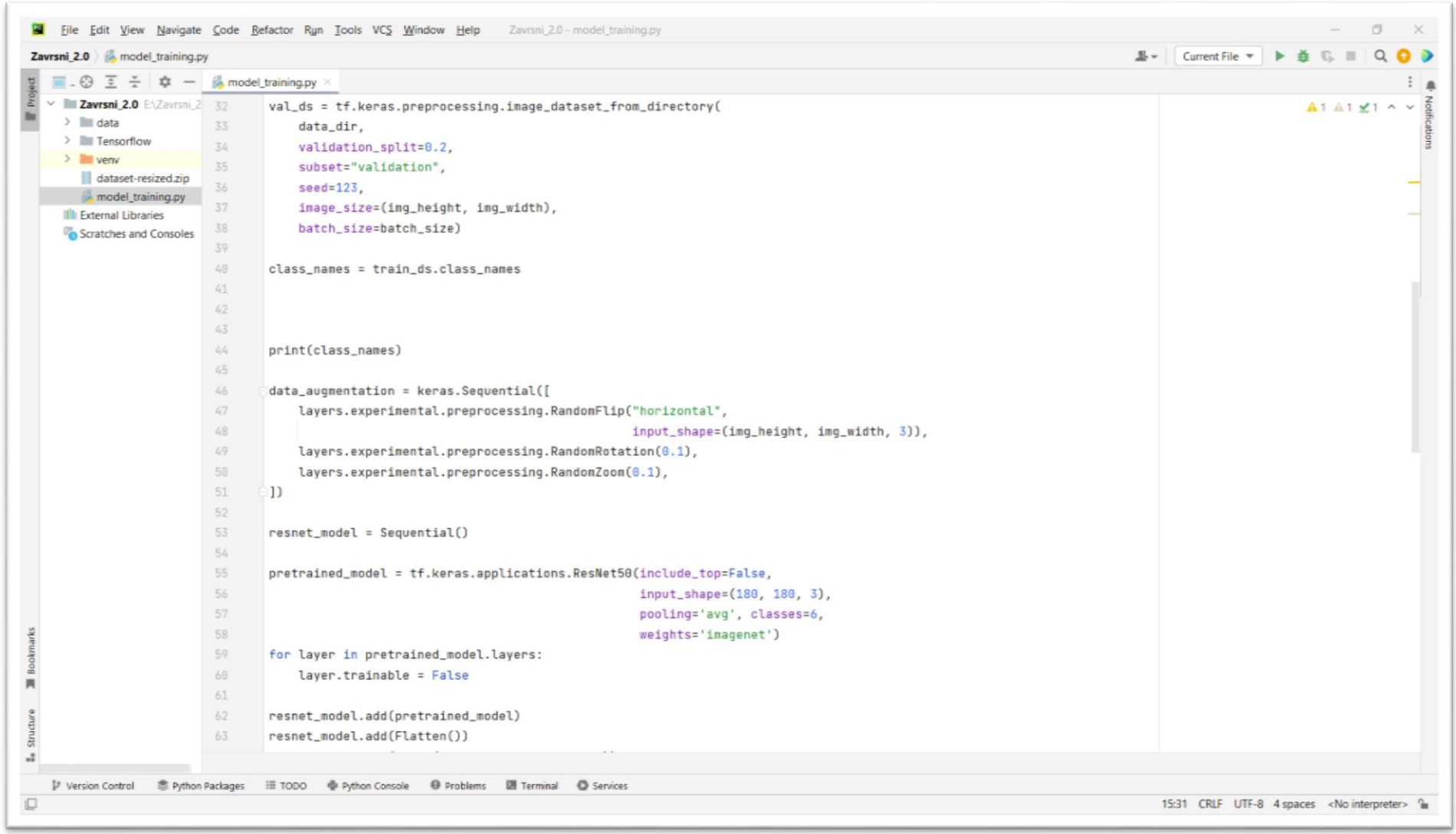
Prilozi

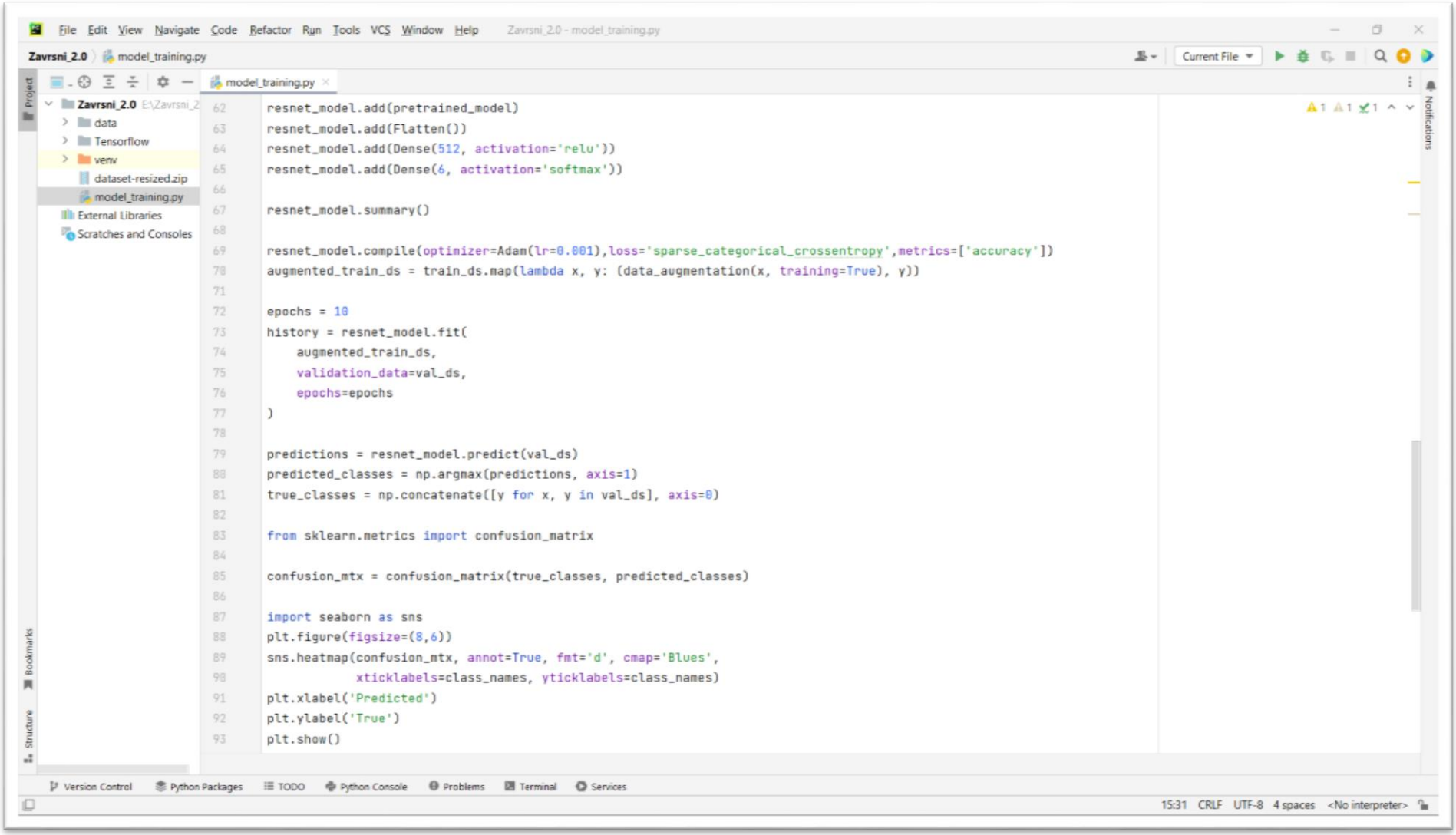
I: Programski kod

Prilog I: Programski kod

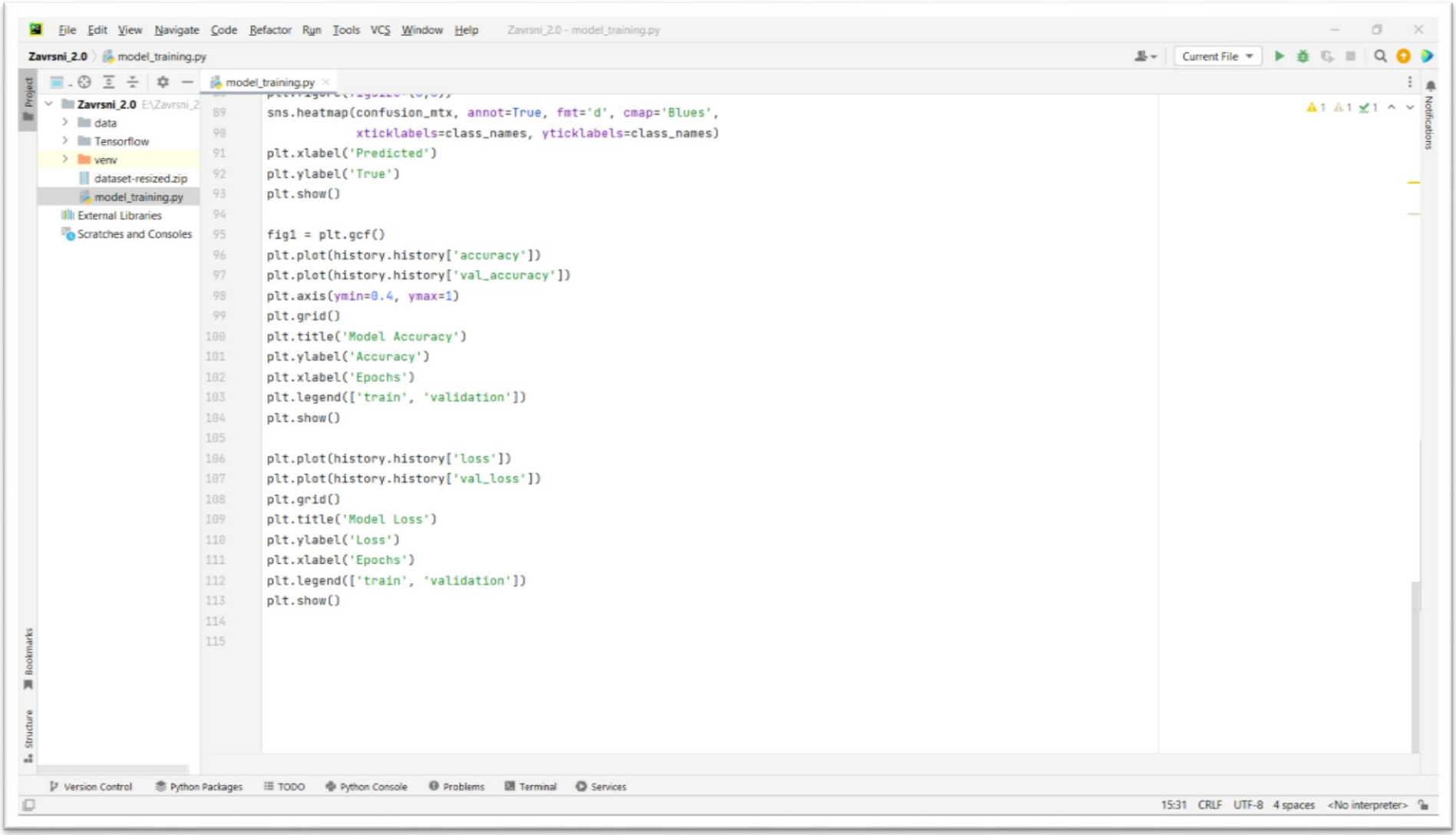


```
import zipfile as zf
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
from keras import layers
from tensorflow.python.keras.layers import Dense, Flatten
from keras.models import Sequential
from keras.optimizers import Adam
import pathlib
import matplotlib.pyplot as plt

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

data_dir = ("data")

img_height, img_width = 180, 180
batch_size = 32

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(
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Zavrsni_2.0 - model_training.py
Zavrsni_2.0 model_training.py
Current File
Project
Zavrsni_2.0 E:\Zavrsni_2
data
Tensorflow
venv
dataset-resized.zip
model_training.py
External Libraries
Scratches and Consoles
model_training.py
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
print(class_names)
data_augmentation = keras.Sequential([
layers.experimental.preprocessing.RandomFlip("horizontal",
input_shape=(img_height, img_width, 3)),
layers.experimental.preprocessing.RandomRotation(0.1),
layers.experimental.preprocessing.RandomZoom(0.1),
])
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())
Notifications
Bookmarks
Structure
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
15:31 CRLF UTF-8 4 spaces <No interpreter>



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

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

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)

from sklearn.metrics import confusion_matrix

confusion_mtx = confusion_matrix(true_classes, predicted_classes)

import seaborn as sns
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()
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Zavrsni_2.0 - model_training.py
Zavrsni_2.0 model_training.py
Current File
Project
model_training.py
Zavrsni_2.0 E:\Zavrsni_2
data
Tensorflow
venv
dataset-resized.zip
model_training.py
External Libraries
Scratches and Consoles
sns.heatmap(confusion_mtx, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
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()
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()
Notifications
Bookmarks
Structure
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
15:31 CRLF UTF-8 4 spaces <No interpreter>