MODEL ZA KLASIFIKACIJU SLIKA OBUĆE

Stazić, Nikola

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

NIKOLA STAZIĆ

Z A V R Š N I R A D

MODEL ZA KLASIFIKACIJU SLIKA OBUĆE

Split, rujan 2023.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

Predmet: Strukture podataka i algoritmi

Z A V R Š N I R A D

Kandidat: Nikola Stazić

Naslov rada: Model za klasifikaciju slika obuće

Mentor: dr. sc. Toma Rončević, prof. visoke škole

Split, rujan 2023.

SADRŽAJ

Sažetak 1
Summary 1
1 Uvod 2
2 Računalni vid 4
2.1 Digitalna slika 4
2.2 Zadaci računalnog vida 5
3 Strojno učenje 9
3.1 Nadzirano učenje 9
3.2 Nenadzirano učenje 10
3.3 Polu-nadzirano učenje 11
3.4 Pojačano učenje 12
3.5 Prijenosno učenje 12
3.6 Klasifikacijski modeli 13
3.6.1 Support vector machines 15
3.6.2 Random forests 15
3.6.3 Neuronske mreže 15
4 Povećanje podataka 16
5 Neuronske mreže 19
5.1 Konvolucijske neuronske mreže 21
5.2 Python 25
5.3 TensorFlow 25
5.4 VGG16 model 27
6 Praktičan rad 29
6.1 Priprema podataka 30
6.2 Izrada modela 35
7 Zaključak 49
Literatura 51

○ SAŽETAK

U ovom završnom radu obrađeni su ključni koncepti i tehnike potrebne za pripremu, izradu, treniranje i validaciju modela klasifikacije slika, izgrađenih uz pomoć konvolucijskih neuronskih mreža. U ovom radu su objašnjeni osnovni koncepti, tehnike i prakse povezane sa strojnim učenjem i računalnim vidom, uključujući i praktični rad koji koristi TensorFlow biblioteku za Python. Praktični rad je izveden na platformi Google Colaboratory u nekoliko koraka: prikupljanje slika, obrada slika te nekoliko ciklusa izrade, treniranja i testiranja modela klasifikacije slika.

Ključne riječi: *klasifikacija slika, konvolucijske neuronske mreže, prijenosno učenje TensorFlow, Python*

○ SUMMARY

FOOTWEAR IMAGE CLASSIFICATION MODEL

This paper presents key concepts and techniques needed to prepare, create, train and validate image classification models, built using convolutional neural networks. The goal of this paper is to explain in short terms basic concepts, techniques and practices connected to machine learning and computer vision, including a practical work that uses TensorFlow library of Python. Practical work was done in Google Colaboratory platform in multiple steps: acquisition of images, image processing and several circles of creating, training and testing image classification model.

Keywords: *image classification, convolutional neural networks, transfer learning, TensorFlow, Python*

1 UVOD

Digitalne slike postale su neizbježan dio naše svakodnevice. One se koriste u različitim područjima, uključujući medicinu, sigurnost, transport i zabavu. U medicini, digitalne slike se koriste za dijagnosticiranje bolesti, praćenje napretka terapija i proučavanje ljudskog tijela. U sigurnosti, digitalne slike se koriste za prepoznavanje lica, prepoznavanje registarskih oznaka i nadzor nad objektima. U transportu, digitalne slike se koriste za nadzor nad prometom, navigaciju i upravljanje vozilima. U zabavi, digitalne slike se koriste u filmovima, video igrama i virtualnoj stvarnosti.

Računalni vid (engl. *computer vision*) je znanstveno područje koje se bavi analizom, obradom i tumačenjem digitalnih slika i videa. Cilj računalnog vida je razumjeti i interpretirati sadržaj slika, na primjer prepoznavanje objekata, lica i gesta, kako bi se omogućilo računalima da oponašaju ljudski vid i da obavljaju zadatke koji su prije mogli izvršavati samo ljudi.

Jedan od ključnih zadataka računalnog vida je klasifikacija slika (engl. *image classification*). To je proces klasifikacije digitalnih slika u različite kategorije na temelju karakteristika sadržaja, kao što su oblik, boja i tekstura. Klasifikacija slika omogućuje računalima da razumiju i kategoriziraju sadržaj slika, što je ključno za mnoge primjene računalnog vida.

U ovom radu će biti predstavljeni neki od ključnih tehnika strojnog učenja (engl. *machine learning*) koje se koriste u računalnom vidu, uključujući nadzirano učenje (engl. *supervised learning*), nenadzirano učenje (engl. *unsupervised learning*), polu-nadzirano učenje (engl. *semi-supervised learning*), učenje potkrepljivanjem (engl. *reinforcement learning*) te o prijenosnom učenju (engl. *transfer leaning*). Prijenosno učenje, naročito korisna tehnika u izradi ovoga rada, je tehnika koja se koristi u klasifikaciji slika, a omogućuje prijenos znanja s jednog problema klasifikacije slika na drugi sličan problem.

Uz to, ovaj rad predstavlja konvolucijske neuronske mreže, jednu od najpopularnijih tehnika za klasifikaciju slika. Konvolucijske neuronske mreže su vrsta neuronskih mreža koje koriste konvolucijske slojeve za automatsko izdvajanje značajki iz slika. Ove mreže postigle su izvanredne rezultate u klasifikaciji slika u usporedbi s drugim tehnikama strojnog učenja.

Kako se digitalne slike danas koriste u različitim područjima, njihova točna i brza klasifikacija od velike je važnosti. U tom kontekstu, strojno učenje i računalni vid postali su vrlo važni za razvoj algoritama koji omogućuju točnu i pouzdanu klasifikaciju slika.

Programski jezik Python i biblioteka TensorFlow su korišteni prilikom implementacije, a VGG16 model korišten je kao bazni model za izgradnju izvornih modela klasifikacije slika.

U praktičnom dijelu rada, korištena je platforma Google Colaboratory s ciljem implementacije modela i testiranja performansi na različitim skupovima podataka. Praktični rad se sastojati od nekoliko faza koje uključuju pripremu slika i nekoliko ciklusa izrade, treniranja i testiranja modela klasifikacije slika.

Cilj ovog rada je da primjenom i objedinjavanjem različitih tehnika u jedinstvenu svrhu dati doprinos daljnjem razvoju algoritama za klasifikaciju slika u širokom rasponu primjena.

2 RAČUNALNI VID

Računalni vid [1]-[2] je znanstveno područje, klasificirano kao dio umjetne inteligencije, koje definira metode kojima se prikupljaju, analiziraju, obrađuju i interpretiraju slike iz realnog svijeta, kako bi se iste pretvorile u računalu razumljive podatke. Drugim riječima računalni vid je skup zadataka koji za rezultat daju računalu razumljivu podatke iz slike.

Razna područja se oslanjaju na računalni vid, kao na primjer obrada slika (engl. *image processing*) i prepoznavanje uzorka (engl. *pattern recognition*). Premda se oba područja oslanjaju na računalni vid znatno se razlikuju te ih se može kategorizirati po zadatku koji izvršavaju. Obrada slika, izvršava zadatak obrade signala (engl. *signal processing*), koji pretvara ulaznu sliku u željenu izlaznu primjenjujući filtere za smanjenje šuma (engl. *noise reduction*), pojačavanje kontrasta (engl. *contrast enhancement*) i regresije (engl. *regression*). Dok prepoznavanje uzoraka spada u klasifikacijski zadatak, odnosno svrstavanje slika u predefinirane kategorije.

Jedan od zadataka računalnog vida je razumijevanje slike, koje kreira opis, ne samo slike, već trodimenzionalne scene na slici, koje računalo razumije.

2.1 DIGITALNA SLIKA

Digitalna slika [1], [3] se sastoji od konačnog skupa digitalnih elemenata koji se nazivaju pikseli, kojima prikazujemo dvodimenzionalne slike. Svaki piksel predstavlja numeričku vrijednost intenziteta svjetlosti ili sive boje. Digitalna slika, kao izraz, najčešće se odnosi na rasterske digitalne slike (engl. *Bitmapped image; Raster image*), što znači da digitalna slika ima fiksnu razlučivost. Digitalna slika s promjenjivom vrijednosti razlučivosti naziva se vektorska digitalna slika (engl. *Vector image*). Danas veliki broj grafičkih sustava koristi kombinaciju rasterske i digitalne slike.

Piksel [3] je najmanji gradivni element digitalne rasterske slike, svaki piksel ima definiranu boju, korištenjem tri ili četiri sastavnice intenziteta, kao što su crvena, zelena i plava ili cijan, magenta, žuta i crna. Pikseli u računalnoj memoriji su najčešće pohranjeni kao rasterska digitalna slika, odnosno dvodimenzionalni niz malih cjelobrojnih vrijednosti. Razlika u pikselima definira se količinom bitova koje piksel koristi kako bi definirao boju

(engl. *bits per pixel*) i označava se sa bpp. Najmanji prikaz boje pikselima ostvaruje se s jednim bitom, koji se naziva jednobojni piksel (engl. *Monochrome pixel*). Svaki dodatni biti za zapis boje, udvostručuje broj boja koje se mogu definirati. Tako na primjer 2 bpp piksel može definirati četiri boje, 3 bpp osam boja. Neke od standardnih mjera su visoka definicija boje (engl. *Highcolor*) sa 16 bpp ili stvarna, odnosno realna, definicija boje (engl. *Truecolor*) s 24 bpp.

Rasterska digitalna slika predstavlja dvodimenzionalnu sliku korištenjem konačnog broja elemenata pravokutne matrice (engl. *Rectangular matrix*) ili mreže (engl. *Grid*) piksela, najčešće pohranjene u računalnoj memoriji u komprimiranom (engl. *Compressed*) obliku. Standardizirani oblici za zapis slika uključuju JPEG, GIF, PNG i BMP, od kojih samo BMP nije komprimiran. Za veliki broj slika, kompresija ne predstavlja problem, međutim kada su potrebni najmanji detalji na slikama, koji bi nestali prilikom izvršavanja algoritama za kompresiju, za zapis slika se koriste neobrađeni format za slike (engl. *Raw image format*). Ovaj format slike odnosi se na slike koje su minimalno obrađene da bi se mogle zapisati u digitalnom obliku. Rasterske slike su upotrebljavaju u fotografiji i daljinskom očitanju.

Vektorska digitalna slika [3] predstavlja dvodimenzionalnu sliku. Ona nije definirana pomoću konačnog broja elemenata, već pomoću matematičkih funkcija. Ove funkcije definiraju geometrijske oblike koji se nalaze u kartezijevom koordinatnom sustavu. Neki od korištenih geometrijskih oblika su točke, pravci, krivulje i poligoni, najčešći format za zapis vektorskih digitalnih slika je SVG. Ovaj tip slike se koristi kada se kompleksne informacije mogu raščlaniti na jednostavne geometrijske oblike i kada slika treba imati veliku geometrijsku preciznost. Najveća primjena nalazi se u tehničkim granama, kao što su arhitektura, topologija i trodimenzionalno prikazivanje.

2.2 ZADACI RAČUNALNOG VIDA

U ovom poglavlju, će biti opisani zadaci računalnog vida [3]-[6]. Ovakvih zadataka ima mnogo. Ako ih se svede na bazične oblike, tada ih se može svrstati u svega osam osnovnih zadataka. Svaki od ovih osnovnih zadataka rješava specifičan problem, odnosno zadatak. Iako ih se naziva osnovnim zadacima, bilo bi netočno reći kako nisu povezani. Naime neki od ovih osnovnih zadataka oslanjaju se na druge osnovne zadatke, odnosno njihove principe rada te njihova saznanja.

Osnovni zadaci računalnog vida su klasifikacija slika, detekcija objekata, segmentacija slike, generiranje slika, povećanje rezolucije slika, procjena poze, prepoznavanje znakova i video analiza.

Klasifikacija slika [6]-[7] je zadatak u kojem se algoritam trenira da dodijeli oznaku slikama iz predefiniranog skupa kategorija. Na primjer, klasifikator slika može biti treniran da prepoznaje različite vrste životinja, poput mačaka, pasa i ptica. Za zadanu sliku, klasifikator bi analizirao i predvidio kojoj kategoriji slika pripada. Klasifikacija slika je uobičajen zadatak u računalnom vidu i često se koristi u aplikacijama poput prepoznavanja objekata i autonomnih vozila.

Detekcija objekata [3], [6] (engl. *Object detection*) je zadatak u računalnom vidu koji uključuje identifikaciju i lociranje objekata u slikama i videima. To je zahtjevniji zadatak od klasifikacije slika, jer zahtijeva ne samo prepoznavanje da je objekt prisutan u slici, već i određivanje njegovog položaja u slici. Postoji mnogo algoritama i tehnika za detekciju objekata, uključujući korištenje pristupa strojnog učenja poput konvolucijskih neuronskih mreža. Detekcija objekata ima širok raspon primjena, uključujući samovozeća vozila, robote i sigurnosne sustave.

Segmentacija [3], [5] (engl. *Segmentation*) slike je zadatak podjele slike u više segmenta ili regija, svaki od kojih odgovara drugom objektu ili pozadini. To je ključna faza u mnogim lancima obrade slike i često se koristi za izolaciju i analizu određenih objekata ili regija od interesa unutar slike. Postoji mnogo pristupa segmentaciji slika, uključujući ručne pristupe (poput crtanja okvirne linije oko objekta) i automatizirane pristupe koji koriste algoritme strojnog učenja. Segmentacija je važan zadatak u računalnom vidu, jer omogućuje detaljniju analizu slika i može se koristiti za poboljšanje performansi drugih zadataka, poput detekcije objekata i klasifikacije.

Generiranje slika [3], [6] (engl. *Image generation*) je zadatak stvaranja novih, sintetičkih slika koje su slične datom skupu slika za treniranje. Često se koristi u strojnom učenju kao način proširivanja podataka (engl. *data augmentation*) za treniranje modela ili za generiranje sintetičkih podataka za treniranje modela kada stvarni podaci nisu dostupni. Pristupa generiranju slika ima nekoliko, uključujući korištenje generativnih modela poput generativnih protivničkih mreža (engl. *Generative Adversarial Networks*) ili varijabilnih auto-enkodera (engl. *Variational Autoencoders*). Generiranje slika može biti izazovan zadatak, jer zahtijeva od modela da nauči temeljne uzroke i strukture prisutne u podacima

za treniranje, a zatim iskoristi to znanje za generiranje novih, realističnih slika. Ima širok raspon primjena, uključujući računalnu grafiku, proširivanje podataka i umjetničke aplikacije.

Povećanje rezolucije slika, odnosno super-rezolucija slike (engl. *Image super-resolution*), je zadatak generiranja visoko-rezolucijske verzije slike iz ulazne slike niske rezolucije. Često se koristi za poboljšanje vizualne kvalitete slika ili za povećanje rezolucije slika koje su preniska za efektivnu upotrebu. Postoji nekoliko pristupa super-rezoluciji slika, uključujući korištenje algoritama strojnog učenja poput konvolucijskih neuronskih mreža. Povećanje rezolucije slike može biti izazovan zadatak, jer zahtijeva od modela da nauči detalje i strukture prisutne u visoko-rezolucijskim slikama, a zatim iskoristi to znanje za generiranje više rezolucije verzije dane slike niske rezolucije. Ima širok raspon primjena, uključujući poboljšanje kvalitete slika za prikaz ili ispis i povećanje rezolucije slika za daljnju analizu.

Procjena položaja objekta [3] (engl. *Object pose estimation*) je zadatak procjene 3D položaja (tj. položaja i orijentacije) objekta ili osobe iz slike ili skupa slika. To je važan zadatak u računalnom vidu i ima širok raspon primjena, uključujući proširenu stvarnost, robote i interakciju čovjek-računalo. Postoji nekoliko pristupa procjeni položaja, uključujući korištenje algoritama strojnog učenja. Procjena položaja može biti izazovan zadatak, jer zahtijeva od modela da nauči temeljnu 3D strukturu objekta ili osobe, a zatim iskoristi to znanje za procjenu položaja iz 2D slike.

Prepoznavanje znakova [3]-[4], odnosno optičko prepoznavanje znakova (engl. *Optical Character Recognition*, skraćeno *OCR*), je zadatak izdvajanja teksta iz slika i videa. To je važan zadatak u računalnom vidu i ima širok raspon primjena, uključujući digitalizaciju dokumenata, automatski prijevod i automatsko titlovanje slika i videa. Ovi algoritmi analiziraju sliku ili video i identificiraju prisutan tekst u njemu, a zatim pretvaraju tekst u oblik čitljiv za računalo. Postoji mnogo pristupa prepoznavanju znakova, uključujući korištenje algoritama strojnog učenja poput konvolucijskih neuronskih mreža. Prepoznavanje znakova može biti izazovan zadatak, jer zahtijeva od modela prepoznavanje i interpretaciju širokog raspon stilova i fontova teksta, kao i rukopisa koji se razlikuje za svakog čovjeka.

Analiza videa [4] (engl. *Video Analysis*) je proces izdvajanja informacija iz podataka videa. To je široko područje koje obuhvaća različite zadatke, uključujući detekciju objekata,

praćenje objekata, prepoznavanje aktivnosti i razumijevanje scene. Analiza videa je važan zadatak u računalnom vidu i ima širok raspon primjena, uključujući nadzor, robote i zabavu. Analiza videa može biti izazovan zadatak, jer zahtijeva od modela da može analizirati velike količine podataka u stvarnom vremenu i donositi odluke na temelju izdvojenih informacija.

3 STROJNO UČENJE

Strojno učenje [8]-[10] je podskup umjetne inteligencije koji uključuje razvoj algoritama i statističkih modela koji mogu učiti iz podataka i predvidjeti podatke. Temelji se na ideji da sustavi mogu učiti iz podataka, prepoznavati uzorke i donositi odluke s minimalnim ljudskim sudjelovanjem.

Postoji nekoliko vrsta strojnog učenja, nadzirano učenje, nenadzirano učenje, polu-nadzirano učenje, pojačano učenje i prijenosno učenje.

Strojno učenje ima širok raspon primjena, uključujući prepoznavanje slika i govora, obradu prirodnog jezika [4] i predviđanje modeliranja. Ima potencijal da revolucionizira mnoge industrije i sektore, uključujući prijevoz, zdravstvo i proizvodnju.

3.1 NADZIRANO UČENJE

Nadzirano učenje [4], [8]-[12] je vrsta strojnog učenja u kojoj se modelu daju označeni primjeri za treniranje, gdje je za svaki primjer u skupu podataka za treniranje naveden i točan izlaz. Cilj je da model nauči prepoznavati značajke podataka koje su povezane s odgovarajućim izlazom, tako da može donositi točna predviđanja za nove, nepoznate podatke.

U nadziranom učenju, model se trenira na skupu podataka s označenim primjerima, a zatim se testira na odvojenom skupu podataka kako bi se evaluirala njegova performansa. Točnost modela se mjeri pomoću metrika poput točnosti, preciznosti i F-mjere. Ako model ne postiže zadovoljavajuću točnost na testnom skupu podataka, može se poboljšati treniranjem na većem skupu podataka ili promjenom hiper-parametara modela.

Nadzirano učenje se može koristiti za vrlo specifične zadatke, poput dijagnosticiranja bolesti na temelju medicinskih podataka ili predviđanja vremena za odlazak na posao na temelju podataka o prometu. Jedna od glavnih prednosti nadziranog učenja je njegova sposobnost da donosi točna i pouzdana predviđanja uz odgovarajuće treniranje i testiranje. Za uspješno korištenje nadziranog učenja potrebni su označeni podaci za treniranje. Postoji nekoliko vrsta modela koji se koriste u nadziranom učenju, uključujući linearne regresije, k-nn klasifikatore, stabla odlučivanja i neuronske mreže. Odabir odgovarajućeg modela ovisi o karakteristikama podataka i ciljnom zadatku.

Nadzirano učenje je također pogodno za kontinuirano poboljšanje modela. Model se može ažurirati s treniranjem na novim podacima ako se preciznost predviđanja. To može biti korisno za zadatke gdje se podaci stalno mijenjaju, kao što je u slučaju prometnih ili ekonomskih podataka. Nadzirano učenje se često kombinira s drugim vrstama strojnog učenja, kako bi se poboljšala performansa modela. Na primjer, može se koristiti nenadzirano učenje za prvo grupiranje podataka i zatim nadzirano učenje za klasifikaciju grupa.

3.2 NENADZIRANO UČENJE

Nenadzirano učenje [4], [11], [13] je jedna od vrsta strojnog učenja koja se koristi za prepoznavanje značajki i strukture podataka bez ikakvih vanjskih informacija. Ova tehnika se često koristi za detekciju anomalija, procjenu gustoće i kompresiju podataka, kao i za grupiranje sličnih točaka podataka zajedno radi klasifikacije korisnika web stranica i svrstavanja proizvoda u skupine na temelju njihovih svojstava. Nenadzirano učenje se također može koristiti za smanjenje broja atributa u velikim skupovima podataka kako bi se olakšala obrada i izvođenje statističkih analiza.

Vrste modela koji se koriste u nenadziranom učenju, uključuju k-središnje grupiranje, grupiranje pomoću gustoće i grupiranje pomoću latentnih varijabli. Odabir odgovarajućeg modela ovisi o karakteristikama podataka za treniranje i zadatku. Važno je imati dobre podatke za treniranje i dobro razumijevanje problema koji se pokušava riješiti kako bi se postigla uspješna implementacija nenadziranog učenja.

Postoji nekoliko ograničenja nenadziranog učenja. Kod modela koji su trenirani ovom tehnikom, teško je evaluirati točnost i pouzdanost, jer model nema označene primjere podataka. Rezultate nenadziranog učenja može biti teže interpretirati nego kod nadziranog učenja, jer nema jasno određenih izlaza koje se mogu usporediti s predviđanjima modela. Nenadzirano učenje nije primjenjivo na sve vrste problema, jer mnogi zadaci zahtijevaju poznavanje izlaza kako bi se model mogao trenirati.

Nenadzirano učenje se može kombinirati s drugim vrstama strojnog učenja, poput nadziranog učenja ili pojačanog učenja, kako bi se poboljšala performansa modela. Na primjer, može se koristiti nenadzirano učenje za grupiranje podataka, a zatim nadzirano učenje za označavanje grupa s odgovarajućim izlazima.

Važno je napomenuti da nenadzirano učenje može biti skupo u vidu računalnih resursa i novca te vremenski zahtjevno, posebno za velike skupove podataka. To je zato što model mora analizirati sve točke podataka i njihove odnose, što može zahtijevati puno resursa i vremena. Unatoč ograničenjima, nenadzirano učenje ima širok raspon primjena.

3.3 POLU-NADZIRANO UČENJE

Polu-nadzirano učenje [4], [10] je vrsta strojnog učenja koja uključuje korištenje označenih i neoznačenih podataka za treniranje. Često se koristi kada je skupo ili vremenski zahtjevno označiti veliki skup podataka i može dovesti do poboljšanja performansi u odnosu na nenadzirano učenje.

U polu-nadziranom učenju, model se prvo obučava na malom skupu označenih primjera. Zatim se model koristi za označavanje većeg skupa neoznačenih primjera, a označeni i neoznačeni podaci se koriste za precizno podešavanje modela. Ovo može pomoći u poboljšanju performansi modela iskorištavanjem dodatnih podataka, ali i dalje koristeći preciznije oznake iz manjeg skupa podataka. Postoji nekoliko tehnika koje se mogu koristiti za polu-nadzirano učenje, uključujući samo-obuku (engl. *self-training*), su-obuku (engl. *co-training*) i višestruko učenje (engl. *Multi-View learning*). Odabir tehnike ovisit će o karakteristikama podataka i specifičnom zadatku koji se rješava.

Polu-nadzirano učenje se često koristi za zadatke poput klasifikacije teksta i klasifikacije slika, gdje je skupo ili vremenski zahtjevno ručno označiti veliki skup podataka. Može se koristiti i u situacijama u kojima postoji veliki broj ne označenih podataka i mali dio označenih podataka. Polu-nadzirano učenje je korisno sredstvo za poboljšanje performansi modela strojnog učenja u različitim zadacima.

Polu-nadzirano učenje također ima neka ograničenja. Na primjer, kvaliteta oznaka u manjem skupu podataka određuje performansu modela, tako da je važno da te oznake budu što točnije. Također, količina neoznačenih podataka koja se koristi može utjecati na performansu modela - previše ne označenih podataka može dovesti do smanjenja performansi, dok premalo neoznačenih podataka može ograničiti poboljšanja u performansi.

Važno je dobro razumjeti problem koji se pokušava riješiti i imati dobre podatke za treniranje kako bi se postigla uspješna implementacija polu-nadziranog učenja. Proučavanjem i vježbanjem različitih tehnika i modela te razumijevanjem karakteristika

podataka, stručnjaci u području strojnog učenja mogu razviti vještine potrebne za uspješnu primjenu polu-nadziranog učenja u različitim zadacima.

3.4 POJAČANO UČENJE

Pojačano učenje [4], [11], [13] je oblik strojnog učenja u kojem se agent uči putem interakcije s okolinom i primanja nagrada ili kazni za određene akcije. Cilj je da agent nauči najbolje akcije za poduzimanje u datoj situaciji kako bi nagrada bila što veća.

U pojačanom učenju postoje agent i okolina. Agent predstavlja neki entitet koji djeluje u okolini, a okolina je prostor u kojem agent djeluje. Agent ima nekoliko mogućih akcija koje može poduzeti u okolini, a za svaku akciju dobiva određenu nagradu ili kaznu. Cilj agenta je dobiti što veću ukupnu nagradu tijekom vremena.

Pojačano učenje se temelji na ideji da agent treba izabrati akcije koje će mu dati što više nagrada u budućnosti. Da bi to postigao, agent koristi pojmove poput vjerojatnosti i očekivanja da bi procijenio koje će mu akcije donijeti najveću korist.
U pojačanom učenju postoje tri glavna koncepta:

1. Akcije: Agent ima nekoliko mogućih akcija koje može poduzeti u okolini
2. Stanja: Okolina se mijenja tijekom vremena, a stanje agenta je trenutno stanje okoline
3. Nagrade: Agent dobiva nagrade ili kazne za određene akcije koje poduzima u određenim stanjima.

Pojačano učenje se može primijeniti na različite zadatke, poput igranja računalnih igara, kontrole strojeva i robotskog rukovanja. To je vrlo korisno za situacije u kojima je teško predvidjeti sve moguće varijante okoline ili gdje je teško napisati točan algoritam za rješavanje problema.

3.5 PRIJENOSNO UČENJE

Prijenosno učenje [4], [8], [13] je metoda strojnog učenja u kojoj se model koji je treniran na jednom zadatku koristi kao početna točka za model na drugom, povezanom zadatku. Ovo može biti korisno kada je dostupan ograničeni podatkovni skup za drugi zadatak ili kada je drugi zadatak sličan prvom.

Prijenosno učenje se temelji na ideji da mnogi zadaci imaju neke zajedničke elemente koji se mogu iskoristiti u modelu. Na primjer, ako se model trenira na prepoznavanju pasa i mačaka, taj model će imati neke zajedničke karakteristike koje se mogu iskoristiti za drugi zadatak poput prepoznavanja ptica. Time se smanjuje potreba za velikim podatkovnim skupovima i smanjuje se vrijeme potrebno za treniranje modela.
Prijenosno učenje se može podijeliti u nekoliko vrsta:

1. Prilagodba (engl. *Fine-tuning*): Ovo je proces u kojem se trenirani model malo prilagođava za novi zadatak. To se može postići tako da se neke od slojeva modela slobodno mijenjaju ili ponovno treniraju na novom podatkovnom skupu.
2. Izdvajanje (engl. *Feature extraction*): U ovom pristupu, model se koristi za izdvajanje značajki iz podataka za novi zadatak. Nova značajke se onda koriste u novom modelu za obradu podataka.
3. Pred-treniranje (engl. *Pre-training*): Ovo je proces u kojem se model trenira na velikom podatkovnom skupu za općeniti zadatak, a zatim se taj model prilagođava za specifični zadatak. Ovo je korisno kada je dostupan velik podatkovni skup za općeniti zadatak, ali ne i za specifični zadatak.

Prijenosno učenje je korisna metoda koja može smanjiti vrijeme i troškove potrebne za treniranje modela na novom zadatku. Treba imati na umu da prijenosno učenje može dovesti do smanjenja točnosti ako se model prilagođava previše za novi zadatak ili ako se koristi pogrešan model za transfer.

3.6 KLASIFIKACIJSKI MODELI

Klasifikacija slika [10]-[13] je zadatak strojnog učenja u kojem se koriste podaci o slikama i oznake tih slika kako bi se izgradio model koji može prepoznati određene objekte ili kategorije na slici. To se može koristiti za mnoge različite zadatke, poput prepoznavanja vozila na cesti, identifikacije biljaka ili životinja u prirodi, ili prepoznavanja lica u fotografijama.

Jedna od ključnih stvari kod klasifikacije slika je odabir odgovarajućeg skupa podataka za treniranje i testiranje. To uključuje odabir odgovarajućih slika i oznaka te odabir veličine skupa podataka u odnosu na veličinu i složenost modela. Previše mali skup podataka

može dovesti do loše generalizacije modela za nove podatke, dok previše veliki skup podataka može dovesti do smanjenja performansi. Performanse modela, u ovom kontekstu, odnose se na cijeli proces treniranja modela i preciznosti predviđanja. Neki od mogućih uzroka smanjenja performansi su:

1. Porast u vremenu potrebnom za treniranje modela
2. Porast računalnih resursa koji se koriste
3. Prekomjerno prilagođavanje (engl. *overfitting*) modela - model počinje učiti o nečistoćama, umjesto uzorcima u slikama.
4. Kvaliteta podataka - greške je teže pronaći i ukloniti

Važno je odabrati odgovarajuće metrike za evaluaciju performansi modela. To uključuje odabir metrika koje odgovaraju ciljevima zadatka i koje se mogu lako interpretirati. Na primjer, u zadatku klasifikacije slika, metrike poput točnosti, preciznosti i pokrivenosti mogu se koristiti za procjenu performansi modela.

Postoje različiti algoritmi i tehnike koji se mogu koristiti za klasifikaciju slika, uključujući konvolucijske neuronske mreže, "support vector machines", i "random forests". Odabir odgovarajućeg algoritma ovisit će o specifičnom zadatku i karakteristikama podataka.

Klasifikacija slika se obično koristi u nadziranom učenju, gdje je model treniran na skupu označenih slika i naučen prepoznavati određene kategorije ili objekte. Zatim se model testira na skupu neoznačenih slika kako bi se procijenila njegova performansa. Model se može poboljšati tako da se ponovo trenira na većem skupu označenih podataka ili da se koriste tehnike poput regularizacije (engl. *Regularization*).

Regularizacija, kao pojam, odnosi se na smanjenje složenosti modela dodavanjem ograničenja (engl. *penalty term*) funkciji gubitka.

Klasifikacija slika također može biti izazovna zbog različitih ograničenja, izazova u prikupljanju i označavanju podataka i kvaliteti samih podataka. To uključuje stvari poput neurednih i nečitljivih slika, slika s nedostatkom detalja ili nejasnoćama te slika s različitom svjetlinom i bojama. Razvijanje tehnika za prevladavanje ovih izazova može dovesti do poboljšanja performansi modela.

U konačnici, klasifikacija slika je važno područje strojnog učenja s mnogim primjenama u različitim domenama. S dobrim razumijevanjem različitih algoritama i tehnika

te vježbanjem s različitim skupovima podataka, stručnjaci u području mogu razviti vještine potrebne za uspješnu implementaciju klasifikacije slika u različitim zadacima.

3.6.1 Support vector machines

Support vector machines [4] (SVM) su algoritmi za učenje s potporom koji se mogu koristiti za klasifikaciju slika. Oni koriste razdvajanje podataka u kategorije pomoću hiperravnina, tako da se što veći broj podataka može opisati hiperravninom. Ovo se postiže odabirom hiperravnina tako da je što dalje od najbližih točaka podataka iz različitih kategorija. Nakon što se odabere hiperravnina, mogu se koristiti za klasifikaciju novih podataka tako da se odredi u kojoj je strani hiperravnina. Pokazali su se vrlo uspješnima u klasifikaciji slika, posebno u slučajevima gdje je područje za klasifikaciju složeno ili ima mnogo značajki.

3.6.2 Random forests

Random forests su algoritmi za učenje koji se sastoje od skupa stabala odluka. Stabla za odluke su jednostavni modeli klasifikacije koji se obično sastoje od jednog ili više pitanja koja se postavljaju o podacima, a na temelju odgovara na ta pitanja se odlučuje u kojoj se klasi podaci nalaze. Random forests se sastoje od velikog broja stabala za odluke koji su povezani zajedno, a konačna kategorizacija se određuje na temelju većine odluka stabala za odluke. Pokazali su se vrlo uspješnima u klasifikaciji slika, jer se mogu lako prilagoditi složenim područjima za kategorizaciju i postižu visoku točnost. Također se mogu lako interpretirati, jer se odluke svakog stabla odlučivanja mogu vidjeti posebno.

3.6.3 Neuronske mreže

Neuronske mreže [3], [4] su algoritmi za strojno učenje koji su inspirirani strukturom i funkcijom ljudskog mozga. Sastoje se od međusobno povezanih "neurona" koji obrađuju i prenose informacije. Primjenjive su u mnogim zadatima, među kojima je i klasifikacija slika odnosno prepoznavanje objekata u slici. Više u poglavlju 5.

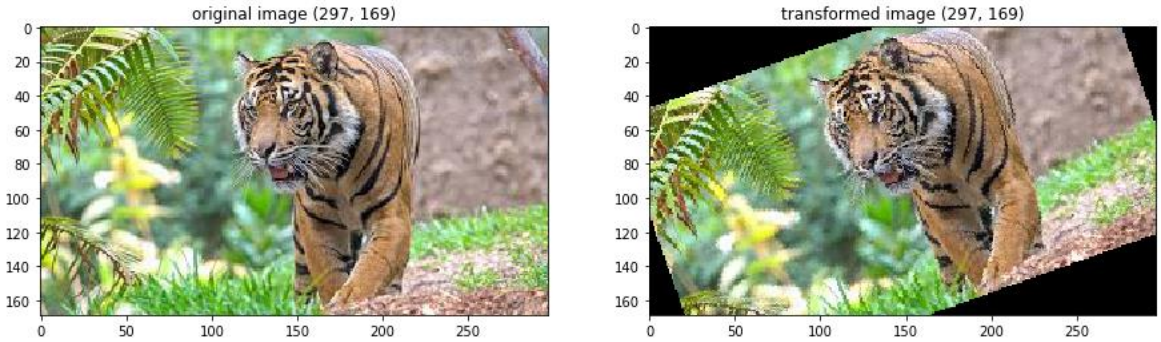
4 POVEĆANJE PODATAKA

Povećanje podataka [3], [4] je proces generiranja novih podataka iz postojećih podataka primjenom skupa slučajnih transformacija. Ove transformacije mogu uključivati stvari poput rotiranja slike, skaliranja u većem ili manjem opsegu, dodavanje šuma ili isječak. Povećanje podataka često se koristi kao način za umjetno povećanje veličine skupa podataka, što može biti korisno prilikom treniranja modela strojnog učenja. Također može pomoći u poboljšanju generalizacije modela što će ga učiniti otpornijim na manje varijacije u podacima.

Ova tehnika može biti korisna u nekim slučajevima, ali to ovisi o tome koji se tip podataka koristi i koja je namjena modela. U nekim slučajevima, povećanje podataka može pomoći u povećanju veličine skupa podataka. Povećanje podataka može povećati vrijeme potrebno za treniranje modela i može dovesti do povećanja složenosti modela. Stoga, treba pažljivo razmotriti je li tehnika vrijedna truda u konkretnom slučaju.

Najčešće korištene metode (tehnike) za povećanje podataka [4] su rotiranje, skaliranje (engl. *scaling*), dodavanje šuma, rezanje (engl. *cropping*), promjena svjetline i promjena boje.

- Rotiranje [3], [4] - proces okretanja slike oko točke za određeni kut (Slika 1), koristi se u svrhu prepoznavanja objekata u različitim položajima.



Slika 1: Rotiranje slike [14]

- Skaliranje [3], [4] - proces mijenjanja veličine slike, povećanje ili smanjenje slike (Slika 2), koristi se u svrhu lakšeg prepoznavanja objekta u različitim veličinama.

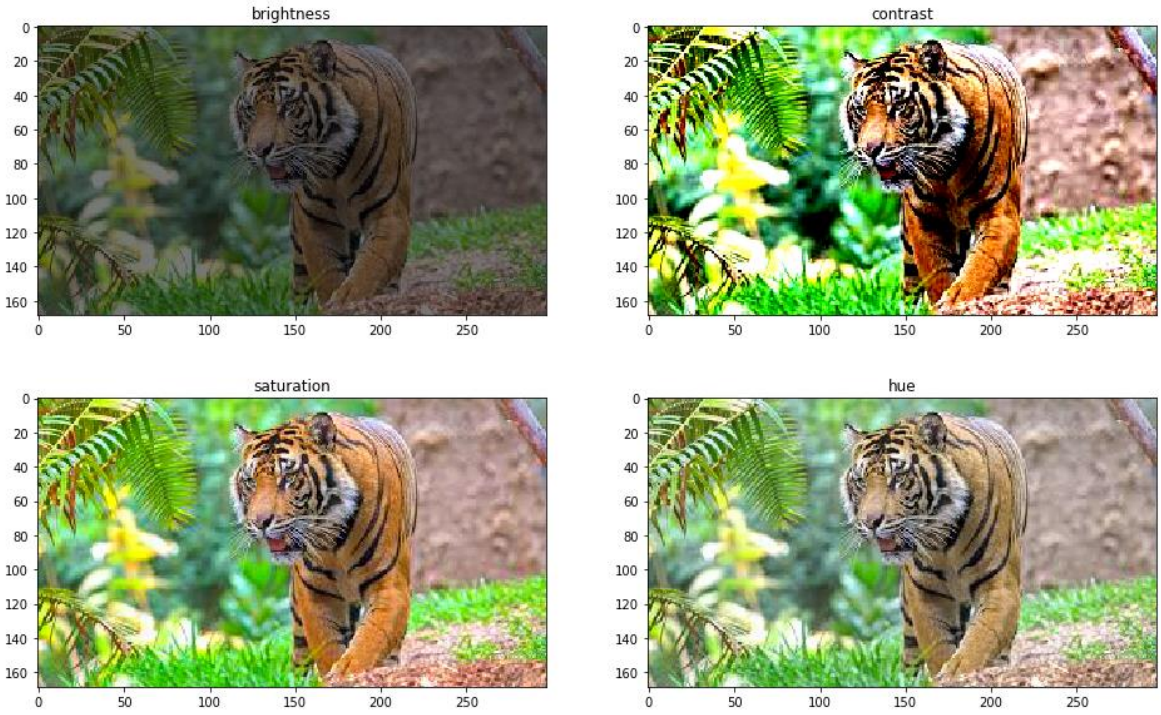
Slika 2: Skaliranje slike [14]

- Rezanje [3] - proces rezanja određenog dijela slike (Slika 3), na primjer uklanjanje drugih objekata iz slike. Neke od prednosti koje ova tehnika može dati su prepoznavanje objekata na rubovima slike i bolje prepoznavanje objekta na slikama s različitim pozadinama.

Slika 3: Rezanje slike [14]

- Promjena svjetline [3], [4] - proces promjene svjetline ili kontrasta slike (Slika 4). Promjena svjetline se postiže mijenjanjem svjetline svakog piksela za istu vrijednost, dok se primjenom ne linearne funkcije postiže promjena kontrasta slike. Ova metoda može povećati preciznost predviđanja modela za slike u različitim osvjetljenjem ili različitih kontrasta.
- Promjena boje [4] - proces mijenjanja boje slike (Slika 4), najčešće se postiže mijenjanjem pojedinačnih kanala boje slike ili primjenom filtera boja. Filteri boja mogu uzrokovati promjenu kontrasta slike. Ova metoda može povećati preciznost predviđanja za slične ili iste objekte u različitim bojama
- Dodavanje šuma - process dodavanja različitih šumova (nečistoća) u slike (Slika 4), šumovi i nečistoće mogu nastati iz različitih razloga među kojima su loši uvjeti

prilikom fotografiranja i greške prilikom obrade digitalne slike. Ova metoda pomaže da model bude otporniji na iste.



Slika 4: Promjena svjetline, boje i zasićenosti slike [14]

5 NEURONSKE MREŽE

Neuronske mreže [3], [4], [15] su algoritmi za strojno učenje koji su inspirirani strukturom i funkcijom ljudskog mozga. Sastoje se od međusobno povezanih "neurona" koji obrađuju i prenose informacije. Koriste se za različite zadatke, poput klasifikacije slika, prijevoda jezika i čak igranja igara. Oni su posebno dobri u prepoznavanju uzoraka i donošenju odluka na temelju podataka, bez eksplicitno navedenih postupaka. Već neuronska mreže pomoću uzoraka i odnosa donosi odluke.

Iako su neuronske mreže postale popularne u posljednje vrijeme, ta tehnologija strojnog učenja postoji već neko vrijeme. Kako su računala postajala sve bolja, a količine podataka veće, neuronske mreže su postale “sposobne” obavljati sve složenije zadatke. Sastoje se od mnogih slojeva “neurona” koji su međusobno povezani. Svaki neuron prima ulazne informacije, a zatim ih obrađuje koristeći određeni matematički model i šalje rezultat dalje. Ovaj proces se ponavlja kroz cijelu mrežu, što omogućuje neuronskoj mreži da se nauči prepoznavati uzorke i odnose u podacima. Postoje različiti tipovi neuronskih mreža, odabir odgovarajućeg tipa ovisi o specifičnom problemu koji se želi riješiti. Na primjer, postoje perceptroni (engl. *perceptrons*) koji se koriste za klasifikaciju, povratne neuronske mreže (engl. *recurrent neural networks*) koje se koriste za obradu teksta i konvolucijske neuronske mreže koje se koriste za obradu slika.

Unatoč mogućnosti postizanje visoke točnosti u različitim zadacima, još uvijek nisu savršene i mogu imati poteškoća s generalizacijom za nove podatke. Primjena neuronskih mreža je široka i može se primijeniti u različitim industrijama. Uključuju prepoznavanje govora, automatizirano prevođenje teksta, klasifikaciju slika i videozapisa, prepoznavanje emocija iz govora, predviđanje vremenskih uvjeta.

Za izradu neuronske mreže, potreban je veliki skup podataka i računalo za obradu istih. Neuronska mreža analizira podatke i “uči” prepoznavati uzorke i odnose. Ovaj proces se naziva “treniranje”. Tijekom treniranja, neuronska mreža se “hrani” velikim brojem primjera, a njeni unutarnji parametri, težine (engl. *weights*) i pristranosti (engl. *biases*), prilagođavaju se kako bi bolje prepoznali uzorke u podacima i njihove odnose. Algoritmi koji se koriste za prilagođavanje nazivaju se optimizatori (engl. *optimizers*), odnosno algoritmi za optimizaciju. Cilj treniranja je pronaći optimalan skup težina koji omogućuje neuronskoj mreži točno klasificiranje ili predviđanje na temelju novih podataka. Nakon što

je neuronska mreža trenirana, može se koristiti za predviđanje ili donošenje odluka na temelju novih podataka. Na primjer, može se trenirati da prepoznaje slike pasa i mačaka, a zatim može klasificirati nove slike kao pse ili mačke. Treniranje neuronskih mreža je zahtjevan proces zbog zahtjeva za velikom količinom podataka i računalnih resursa. Kada je trenirana, neuronska mreža može postići visoku točnost u različitim zadacima. Stoga se može očekivati da će se njihova primjena nastaviti širiti u budućnosti, što bi moglo utjecati na mnoge industrije.

Proces treniranja neuronske mreže odvija se u epohama. Kako bi neuronska mreža mogla vrednovati izlazne rezultate nakon svake epohe te koristiti iste za poboljšanja predviđanja, koriste se algoritmi propagacije greške unazad, odnosno "backpropagation" algoritmi (engl. *Backpropagation algorithm*). U kontekstu strojnog učenja, ovo je skupni naziv za sve algoritme koji se koriste za izračunavanje gradijenta funkcije gubitka (engl. *gradient of loss function*), uzimajući u obzir težine klasa. Algoritmi s povratnim postupkom definiraju samo način na koji se računa gradijent funkcije gubitka, a u praksi se često koristi kao naziv za cijeli proces učenja. Ovaj termin u neuronskim mrežama prvi put se pojavio 1986. godine [16], dok sami princip algoritama s povratnim postupkom se pojavljuje 1957. godine [17].

Pomoću "*backpropagation*" algoritama, pogreška se prenosi "unatrag" kroz mrežu, od izlaznog do ulaznog sloja. U ovom procesu algoritam računa dodatke na parametre koristeći težine i pristranosti mreže. Rezultat ovog algoritma je derivacija funkcije pogreške u odnosu na svaku težinu i pristranost. Temeljem ovog rezultata, težine i pristranosti se ažuriraju koristeći algoritme optimizacije, u svrhu smanjenja pogreške.

Algoritmi optimizacije koriste se u svrhu minimizacije funkcije pogreške. Ovi algoritmi iterativno ažuriraju parametre modela sve dok funkcija pogreške nije minimiziranja. Jedni od najčešće korištenih algoritama za optimizaciju su Adam i SGD.

Adam [4] (engl. *Adaptive moment estimation*, skraćeno Adam) je algoritam za optimizaciju koji kombinira elemente optimizacije temeljem zamaha (engl. *momentum-based optimization*) i adaptivne stope učenja (engl. *adaptive learning rate*). U ovom algoritmu koristi se jednadžba za eksponencijalno smanjenje pomičnih prosjeka (engl. *exponentially decaying moving averages*) i za gradijent i za moment gradijenta drugog reda (engl. *second moments of gradients*). Pomični prosjek (engl. *moving averages*) omogućava

prilagodljivu promjenu parametra učenja, što kao rezultat daje veću brzinu konvergencije i bolje performanse. Ovaj algoritam pripada naprednoj skupini optimizatora.

SGD [4] (engl. *Stochastic Gradient Descent*, skraćeno SGD) je algoritam optimizacije koji ažurira vrijednosti parametara mreže koristeći prosječni gradijent funkcije pogreške nad malim podskup skupa podataka za treniranje. Proces ažuriranja se ponavlja do minimalne konvergencije funkcije pogreške. Ovaj algoritam pripada skupini osnovnih algoritama.

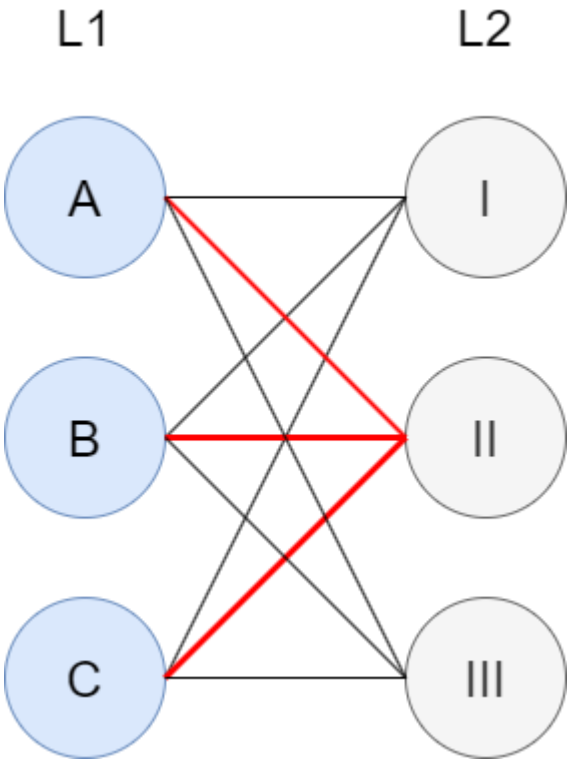
Općenito, Adam se smatra otpornijim, odnosno boljim, algoritmom optimizacije nego SGD, međutim u slučaju ograničenih resursa ili velikih skupova podataka za treniranje, SGD je češće korišten optimizator.

5.1 KONVOLUCIJSKE NEURONSKE MREŽE

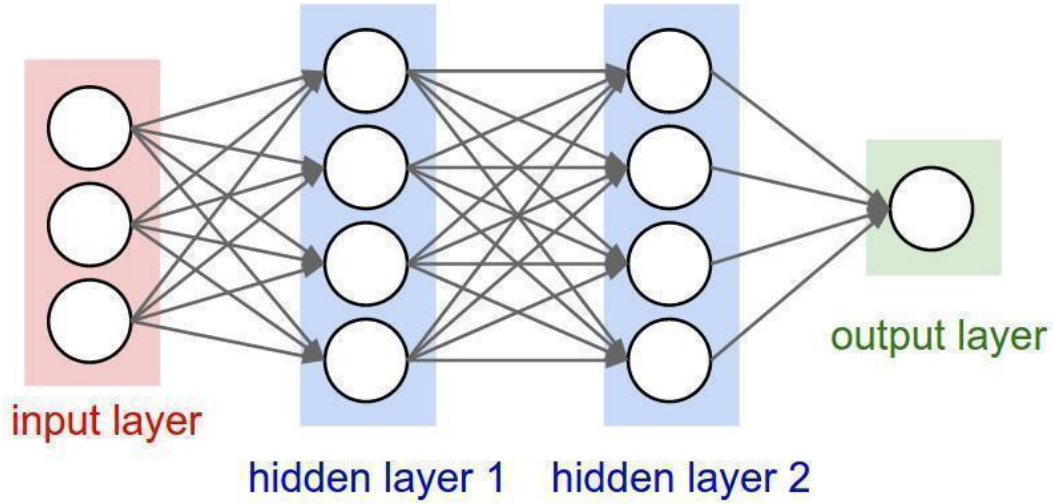
Konvolucijske neuronske mreže [5], [8], [13] su dosta slične običnim neuronskim mrežama, sastoje se od neurona koji imaju promjenjive težine i pristranosti, svaki od neurona prima ulazne podatke nad kojima se vrši skalarni umnožak. Ova mreža i dalje predstavlja samo jednu diferencijalnu funkciju rezultata. Kod konvolucijskih neuronskih mreža ulazni podatak je eksplicitno definiran kao slika. Ova promjena omogućava da je funkcije propusta (engl. *forward functions*), u konvolucijskim neuronskim mrežama, znatno efikasnije implementirati te uvelike smanjuje količinu parametara mreže.

Neuroni u ovim mrežama su raspoređeni u tri dimenzije; širina, visina i dubina. U ovom kontekstu dubina se ne odnosi na dubinu neuronske mreže, odnosno ne odnosi se na ukupan broj slojeva neuronske mreže već na treću dimenziju ulaznog podatka. Na primer za sliku 224x224x3, odnosno za sliku veličine 224x224 piksela, u kojoj je svaki piksel opisan s tri boje, prvi broje je širina, drugi visina, a treći dubina. Neuroni u ovoj arhitekturi nisu potpuno povezani, već je svaki neuron povezan tek s malom regijom neurona u prethodnom sloju. Znatna promjena arhitekture je u izlaznom sloju, kod konvolucijskih neuronskih mreža izlazni sloj je dimenzija 1x1xC, gdje C je broj jednak ukupnom broju klasa.

Potpuno vezani slojevi, kao pojam, označava veze neurona u sloju s prethodnim slojem. Potpuno vezi neuron predstavlja vezu jednog neurona sa svakim iz prethodnog sloja. Troslojna neuronska mreža s potpuno vezanim slojevima prikazana je na slici 6.

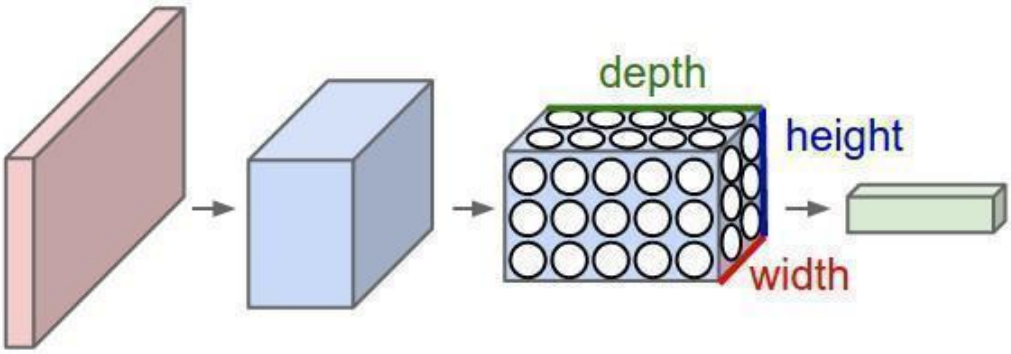


Slika 5: Potpuno vezani slojevi



Slika 6: Troslojna neuronska mreža

Na slici 7 prikazana je konvolucijska neuronska mreža, s neuronima raspoređenim u trodimenzionalne slojeve, ulaz u svaki sloj je trodimenzionalan kao i izlaz. Crveni sloj predstavlja ulazni podatak, odnosno sliku, gdje je treća dimenzija je broj osnovnih boja slike.



Slika 7: Troslojna konvolucijska neuronska mreža

U arhitekturi konvolucijskih neuronskih mreža tri glavne vrste slojeva su konvolucijski sloj (engl. *Convolutional Layer*), sloj za udruživanje (engl. *Pooling Layer*) te potpuno povezani sloj (engl. *Fully-Connected Layer*). Korištenjem ove tri vrste slojeva, aktivacijskog sloja te ulaznog sloja, može se izraditi potpuna konvolucijska neuronska mreža. Kako bi pobliže objasnili što svaki od ovih slojeva radi, za primjer ćemo koristiti najjednostavniju arhitekturu konvolucijskih neuronskih mreža, [INPUT - CONV - RELU - POOL - FC]

INPUT - [224x224x3], odnosno ulazni sloj, je sloj koji sadrži izvorne vrijednosti piksela slike, u ovom slučaju je to slika širine 224, visine 224 i 3 kanala boja.

CONV - konvolucijski sloj, u ovom sloju korištenjem skalarnog produkta težine neurona s lokalnom regijom iz prethodnog sloja, računaju se izlazne vrijednosti neurona.

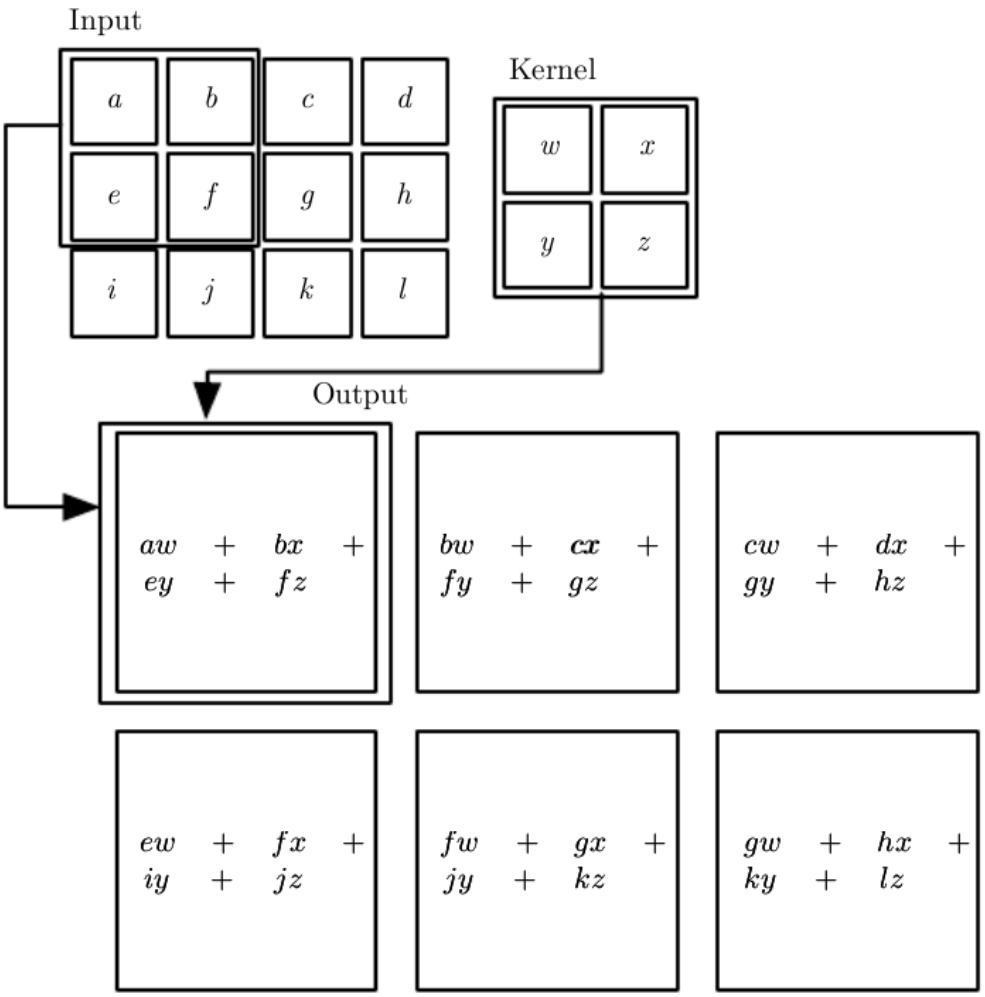
RELU - ReLu sloj aktivacije, će primijeniti funkciju aktivacije na svakom elementu (broju), koja će za izlaz imati 0 ako je element manji ili jednak nuli, a u protivnom vrijednost elementa.

POOL - sloj za smanjenje prostornih dimenzija (engl. *downsampling spatial dimensions*) ulaza, uzimajući najveću ili prosječnu vrijednost regije koja se obrađuje

FC - potpuno povezani sloj, ujedno i izlazni sloj neuronske mreže, služi za izračun vjerojatnosti za svaku kategoriju u kategorizaciji. Ovaj sloj, kako se može zaključiti iz naziva, je primjer obične neuronske mreže, u kojoj je svaki neuron iz ovog sloja povezan sa svakim iz prethodnog sloja.

U sloju CONV/FC za izračun vrijednosti koriste se parametri težine i pristranosti neurona, za razliku od RELU/POOL koji ne ovise o ova dva parametra.

Osnovni sloj konvolucijske neuronske mreže je konvolucijski sloj. Ovaj sloj izvodi operaciju konvolucije (Slika 8). Operacija konvolucije izvodi se nad matricom brojeva, koja se naziva filter ili jezgra (engl. *kernel*) te se prebacuje preko podskupa ulazne matrice ili slike. Vrijednosti u jezgri se množe element po element s odgovarajućim vrijednostima u ulaznoj matrici.



Slika 8: Ilustracija operacije konvolucije [4]

Na slici 8 prikazana je ilustracija operacije konvolucije. Ova operacija predstavlja množenje dijela ulazne matrice (Input) s filter (Kernel) matricom. Ovakvo množenje se odvija sve dok svi redci i stupci ulazne matrice nisu pretvoreni u izlaz (Output).

5.2 PYTHON

Python je programski jezik osmišljen od strane Guido van Rossuma u prosincu 1989. godine. Van Rossum je tada radio u institutu za informatiku u Amsterdamu i bio je frustriran s trenutačnim programskim jezicima koje je koristio za svoj rad, stoga je odlučio napisati svoj vlastiti jezik.

Prva verzija Pythona, verzija 0.9.0, objavljena je 1991. godine. U početku je Python bio popularan među zajednicom akademskih i istraživačkih programera, ali nije bio široko korišten u industriji. To se promijenilo s objavljivanjem verzije 1.0 1994. godine i verzije 2.0 2000. godine, kada je Python postepeno počeo dobivati na popularnosti među profesionalnim programerima. Jedna od glavnih značajki Pythona je njegova "čitljivost" i prijateljstvo prema korisniku (engl. *user friendly*). Python ima jednostavnu sintaksu koja se lako može razumjeti, što ga čini lakšim za početnike. Python također podržava različite paradigme programiranja, uključujući objektno-orijentirano, funkcionalno i proceduralno programiranje.

Python se također može koristiti za različite svrhe, uključujući web development, strojno učenje, analizu podataka i mnogo više. Mnogi veliki tehnološki divovi, poput Googlea, Facebooka i Netflixa, koriste Python u svojim poslovnim procesima. Do danas je objavljeno više od 25 verzija Pythona. Trenutna stabilna verzija je Python 3.10, objavljena u veljači 2021. godine. U budućnosti se očekuje da će Python nastaviti rasti u popularnosti i biti jedan od vodećih programskih jezika u industriji.

5.3 TENSORFLOW

TensorFlow [18] je biblioteka otvorenog tipa (engl. *open source*) koju je napravio Google Brain tim, za strojno učenje i umjetnu inteligenciju. 2015. godine pod licencom Apache Licence 2.0 objavljena je prva verzija biblioteke, poboljšana verzija TensorFlow 2.0 objavljena je 2019. godine. Danas se ova biblioteka može koristiti u mnoštvu programskih jezika, među kojima su Python, JavaScript, C++ i Java te je dostupna na različitim operativnim sustavima kao što su Linux, Windows, Mac OS, Android i IOS. Fleksibilna arhitektura ove biblioteke omogućuje da se za treniranje koriste procesori (engl. *Central Processing Unit*, skraćeno *CPU*), grafičke kartice (engl. *Graphical Processing Unit,*

skraćeno *GPU*) i jedinice za obradu tenzora [4] (engl. *Tensor Processing Unit*, skraćeno *TPU*), na jednom ili skupu (engl. *cluster*) računala.

Matematičke operacije koje TensorFlow izvršava su prikazane kao grafovi toka podataka sa stanjem (engl. *stateful dataflow graphs*), samo ime TensorFlow dolazi od naziva za takve matematičke operacije koje su izvršene na višedimenzionalnom nizu podataka.

Kako bi se poboljšala efikasnost računala koji treniraju i koriste trenirane modele, Google je 2016. godine napravio jedinice za obradu tenzora, isključivo za primjenu u strojnom učenju. Ova tehnologija koristi ubrzanje obrade podataka korištenjem umjetne inteligencije (engl. *AI acceleration*) u svrhu pružanja visoke količine obrade podataka, s aritmetičkim operacijama niske preciznosti. Jedinice za obradu tenzora donijele su napredak u strojnom učenju u vidu smanjenja vremena potrebnog za treniranje modela, a zadržavajući ekvivalentnu količinu ostalih računalnih resursa.

Tablica 1: Usporedba performansi procesorskih jedinica

Procesorska jedinica	**TFLOPS (TeraFLOPS)**
CPU	2
GPU	125
TPU	180
TPUv3	420

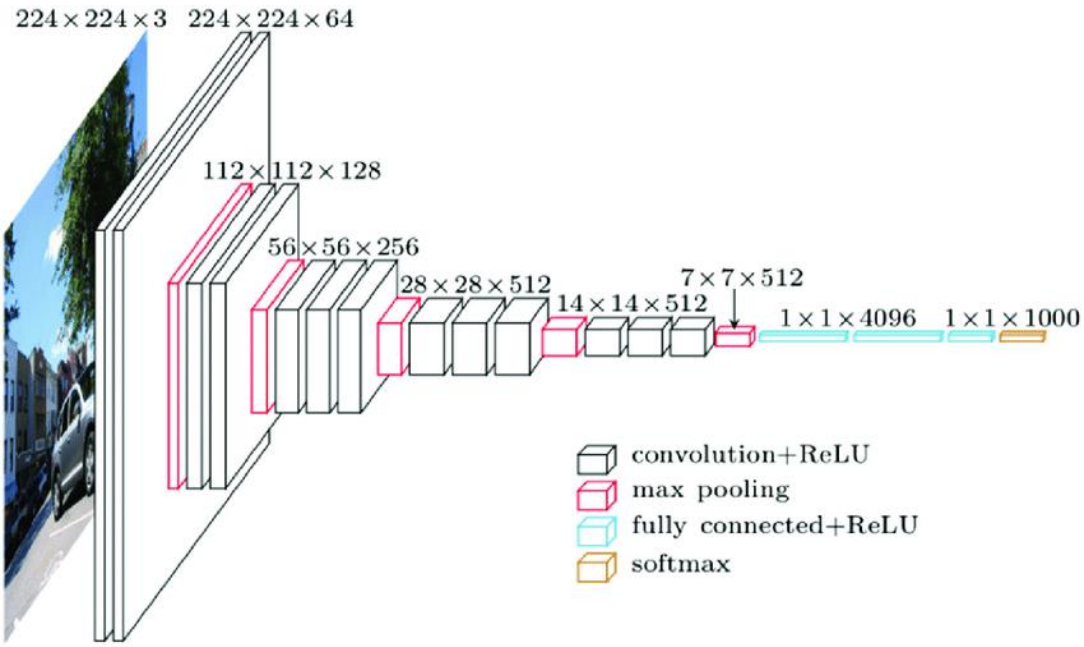
Tablica 1 prikazuje usporedbu [19] procesora (CPU), grafičke kartice (GPU) i jedinice za obradu tenzora (TPU i TPUv3), jedinica za usporedbu performansi je FLOPS, odnosno broj matematičkih operacija nad brojevima s pomičnom točkom u sekundi (engl. *Floating-Point Operations Per Second*). Iako je dosta teško realno usporediti različite vrste procesorskih jedinica, iz tablice je vidljiva velika razlika u brzini obrade podataka, koja je proporcionalna razlici FLOPS-ova dviju procesorskih jedinica.

5.4 VGG16 MODEL

VGG16 [20] je gotovi, odnosno već trenirani model (engl. *pre-trained model*), baziran na konvolucijskim neuronskim mrežama, za prepoznavanje i klasifikaciju objekata u digitalnim slikama. Nastao je 2014. godine na Oxfordu, razvili su ga Simonyan i Zisserman iz Visual Geometry Group, skraćeno VGG, za klasifikacijsko natjecanje u računalnom vidu i prepoznavanju objekata na digitalnim slikama ILSVRC (ImageNet Large Scale Visual Recognition Competition).

Ovaj model treniran je na skupu podataka (engl. *dataset*) od 1,45 milijuna digitalnih slika [20], podijeljenom u tisuću kategorija, sadržanom u tri podskupa, podskup za treniranje (engl. *training dataset*) od 1,3 milijuna digitalnih slika, podskupu za testiranje (engl. *testing dataset*) od sto tisuća digitalnih slika i podskupu za potvrđivanje (engl. *validation dataset*) od pedeset tisuća digitalnih slika. Postignuta preciznost ovog model a je 92,7%. Ovaj model izuzetno je dobar za prepoznavanje pronalaženju svojstava slike (engl. *feature extraction*) što omogućava da se koristi čak i za prepoznavanje objekte koji su nisu dio klasifikacije. Razvijen je u svrhu poboljšanja preciznosti prepoznavanja objekata u digitalnim slikama, povećanjem dubine konvolucijske neuronske mreže.

VGG16 model (Slika 9) sadrži 16 težinskih slojeva (engl. *weight layers*). Ulaz modela je digitalna slika u boji, veličine 224x224 piksela. Slika prolazi kroz više konvolucijskih slojeva s fiksnom veličinom filterom od 3x3 i korakom 1. Model sadrži pet "max pooling" filtera, ugrađenih između konvolucijskih slojeva, kako bi smanjili količinu ulaznih podataka. Nakon konvolucijskih slojeva dolaze tri spojena sloja od 4096, 4096 i 1000 kanala. Zadnji sloj je "soft-max" sloj za polinomnu distribuciju vjerojatnosti (engl. *multinomial probability distribution*). Ovaj model izrazito je uspješan u prepoznavanju objekata, ali preciznost prepoznavanja scene (engl. *scene recognition*) je nešto niža u usporedbi s drugim modelima. Ovaj model osmišljen je za klasifikaciju slika, a može se koristiti i za prijenosno učenje (engl. transfer learning) na drugim zadacima povezanim s obradom slika. VGG16 model jedan je od popularnijih modela za prijenosno učenje zbog svoje jednostavnosti i uspješnosti u mnogim zadacima obrade slika.

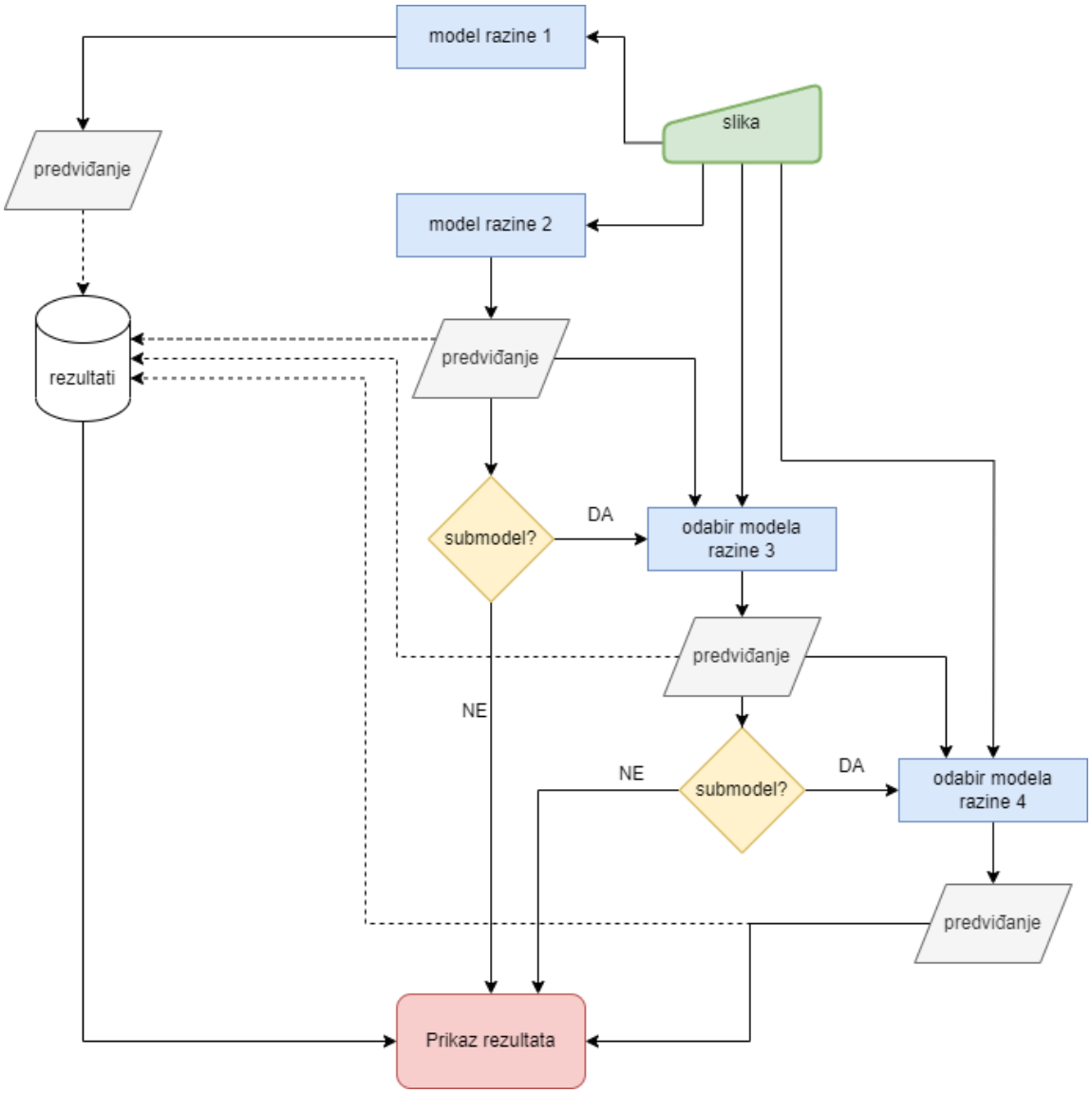


Slika 9: Prikaz arhitekture modela VGG16 [20]

6 PRAKTIČAN RAD

Praktični rad se bazira na višeslojnom klasifikacijskom modelu slika, u kojem su prve dvije razine klasifikacije prisutne za svaki ulazni podatak, a ostale razine nisu obavezne. Kako bi se riješio više jezični problem naziva kategorija, ubrzalo treniranje, odnosno smanjila količina resursa potrebna za treniranje i povećala preciznost predviđanja svi nazivi kategorija su pretvoreni u numeričke vrijednosti.

Za rješavanje višeslojnog klasifikacijskog problema odabran je pristup treniranja zasebnih modela za svaku klasifikacijsku razinu (Slika 10). Praćenjem izlaznih vrijednosti modela određuje se koji model se koristi za određenu klasifikacijsku razinu.



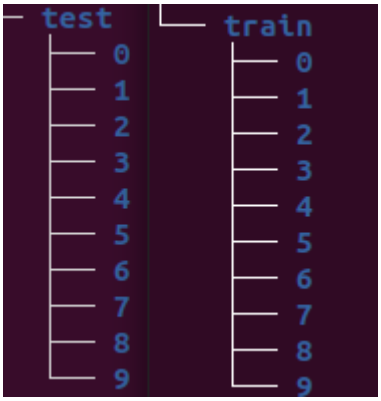
Slika 10: Prikaz procesa predviđanja

Praktični zadatak izvršen je korištenjem google Colaboratory tehnologije, što je omogućava pristup računalnim resursima na oblaku (engl. *cloud*), koja koristi IronPython, građen oko TensorFlow python biblioteke (engl. *library*) i radi na principu virtualnih računala. Prilikom realizacije programskog rješenja kako bi se postigla veća brzina treniranja modela, cijeli skup podataka je kopiran na oblak, te je zbog ograničenosti u prostoru za pohranu podataka, skupa podataka za treniranje modela znatno smanjen.

6.1 PRIPREMA PODATAKA

Proces pripreme podataka, odnosno slika, odvijao se u nekoliko koraka, numeriranje kategorija po klasifikacijskim slojevima, numeričke reprezentacije kategorija za svaku od klasifikacijskih slojeva počinje s 0 i uvećava se za 1 za svaku iduću kategoriju. Kategorije su sortirane po broju slika koje im pripadaju silazno te kategoriji s najvećim brojem slika je dodijeljena kategorija 0, idućoj kategoriji po broju slika 1 itd. U slučaju da postoje dvije kategorije s jednakim brojem slika, koristi se alfabetsko sortiranje uzlazno za odrediti koja kategorija ima manju numeričku vrijednost.

Kako bi se slike pripremile za obradu i treniranje, potrebno je izraditi strukturu direktorija za treniranje i testiranje (Slika 11) te kopirati slike na odgovarajuće lokacije. Ovi direktoriji moraju imati istu strukturu pod direktorija, a imena pod direktorija su dodijeljene cjelobrojne vrijednosti imena kategorija. Struktura je izrađena pomoću bash skripte (Slika 12), broj pod direktorija je varijabilan stoga bash skripta prima broj kategorija kao argument. Slike su kopirane u omjeru 9:1, tj na svakih 9 slika za treniranje, korištena je jedna za testiranje. Dobivena struktura direktorija ručno je kopirana na oblak.



Slika 11: Primjer test i train direktorija

```
#!/bin/bash

if [ $# -eq 0 ]
then
        echo "No arg supplied, requires one argument"
        exit
fi

mkdir -p dataset
mkdir -p dataset/train
mkdir -p dataset/test

for (( dn=0; dn<=${1}; dn++ ))
do
        mkdir -p dataset/train/$dn
        mkdir -p dataset/test/$dn
done
```

Slika 12: Bash skripta za izradu strukture direktorija

Ukupan broj slika koje je bilo moguće kopirati na oblak je 42000 od kojih skup za treniranje sadrži 37800 slika, a skup za testiranje 4200, sve slike su u boji te su veličina 210 x 180 px.

Nakon izrade strukture direktorija, slike su kopirane u odgovarajuće direktorije, a se skup podataka za treniranje modela učitava se koristeći strukturu modela, koristeći funkciju *image_dataset_from_directory* (Slika 13). Obavezni parametri ove funkcije su:

- *directory*: putanja do direktorija koji se učitava
- *batch_size*: veličina skupine podataka, korištena vrijednost
- *image_size*: veličina slike, sve slike će biti promijenjene u ovu veličinu ako već nisu, korištena vrijednost

Osim ovih parametara još korišteni parametri su za učitavanje dvaju skupa podataka iz jednog direktorija, a ti parametri su:

- *validation_split*: vrijednost između 0 i 1, predstavlja dio podataka koji će biti uzet za validacijski skup, korištena vrijednost
- *subset*: definira koji tip podskupa će funkcija vratiti

```
train_ds = image_dataset_from_directory(
    directory=f'{ds_folder}/train', batch_size=batch_size, image_size=image_size,
    validation_split=validation_split, subset='training',
)

valid_ds = image_dataset_from_directory(
    directory=f'{ds_folder}/train', batch_size=batch_size, image_size=image_size,
    validation_split=validation_split, subset='validation',
)

test_ds = image_dataset_from_directory(
    directory=f'{ds_folder}/test', batch_size=batch_size, image_size=image_size,
)
```

Slika 13: Učitavanje podataka iz strukture direktorija

Ostali parametri su korišteni sa zadanim vrijednostima. Neki od tih parametara su:

- *color_mode*: koji definira broj kanala boje, vrijednost: rgb
- *shuffle*: odnosno miješanje redoslijeda slika, vrijednost: True
- *labels*: za stvaranje naziva kategorija iz imena pod direktorija, vrijednost: inferred
- *label_mode*: koji definira koji tip podatka će biti ime kategorije, što direktno utječe na funkciju gubitka modela koja će biti korištena, vrijednost: int
- *interpolation*: definira metodu koja će biti korištena za promjenu veličine slika, vrijednost: bilinear

Prilikom pripreme slika, slike su standardizirane na veličinu 210x180 korištenjem skaliranja, prije toga iz slika su izrezani samo dijelovi koji predstavljaju traženi objekt (Slika 19-20). U prvim testnim podacima također su korištene metode promjene svjetline i boje (Slika 21), čiji rezultati su uklonjeni zbog znatnog povećanja vremena treniranja, zbog višestrukog povećanja broja slika u skupu za treniranje. U nastavku se nalaze primjeri slika iz modela za treniranje.

Slika 14: Kategorija “košulje”

Slika 15: Kategorije “čizme”

Slika 16: Kategorije “tenisice”

Slika 17: Kategorije “majice”

Slika 18: Kategorije “hlače”

Slika 19: Originalna slika

Slika 20: Rezanje slike (lijevo) i smanjenje slike (desno

Slika 21: Promjena boje i svjetline

6.2 IZRADA MODELA

Proces izrade modela odvijao se u nekoliko koraka. Prvi korak bio je odrediti na koji način će se pristupiti izradi modela, izradi "od nule", bez baznog gotovog modela ili koristeći neki od postojećih modela kao osnovu za izradu, odnosno koristeći tehniku prijenosnog učenja. Zbog malog broja slika na kojem se model trenira, model koji je izrađen bez nadogradnji na postojeće, nije postigao zadovoljavajuću preciznost, stoga kao bazni model odabran je VGG16 (Ispis 1). Ovaj bazni model odabran je zbog velike preciznosti koju je postigao i broja ulaznih podataka na kojima je treniran. Model je korišten s imagenet pred treniranim podacima i isključeno je treniranja modela i isključeni su potpuno vezani slojevi modela koristeći parametar "*include_top*" te koristeći parametar "*input_shape*" definirana je veličina slike 210x180 te broj kanala 3.

```
base_model = VGG16(weights=”imagenet”, include_top=False, input_shape
=(210, 180, 3))
base_model.trainable = False
```

Ispis 1: Učitavanje baznog modela

U drugom koraku izrade modela trebalo je odrediti optimalnu strukturu i broj blokova koji bi se koriste kao nadogradnja na bazni model (Ispis 1), jedan od najvažnijih parametara koji se pratio prilikom izrade bloka je vrijeme koje je potrebno za izvršavanje samog bloka, zbog vremenskog ograničenja u ukupnoj dužini treniranja modela, koje na Google Colaboratory platformi iznosi osam sati. Direktan utjecaj na strukturu bloka također ima i broj takvih blokova koji će se koristiti, kao i gustoća svakog gustog sloja, što je pronalazak optimalnih vrijednosti znatno otežalo. Glavno mjerilo za odabir ovih parametara je preciznost modela na testnim podacima, ali na odabir također je i utjecala vrijednost gubitka modela (engl. *model loss*) na skupu testnih podataka kao i na skupu validacijskih podataka.

```
blocks = {
  1: [
    layers.Dense(4096, activation='relu', name="block1_dense1"),
    layers.Dense(4096, activation='relu', name="block1_dense2"),
    layers.Dropout(0.5, name="block1_dropout1"),
  ],
  2: [
    layers.Dense(2048, activation='relu', name="block2_dense1"),
    layers.Dense(2048, activation='relu', name="block2_dense2"),
    layers.Dropout(0.5, name="block2_dropout1"),
  ],
  3: [
    layers.Dense(1000, activation='relu', name="block3_dense1"),
    layers.Dense(1000, activation='relu', name="block3_dense2"),
    layers.Dropout(0.5, name="block3_dropout")
  ]
}
```

Ispis 2: Blokovi i strukture pojedinih blokova

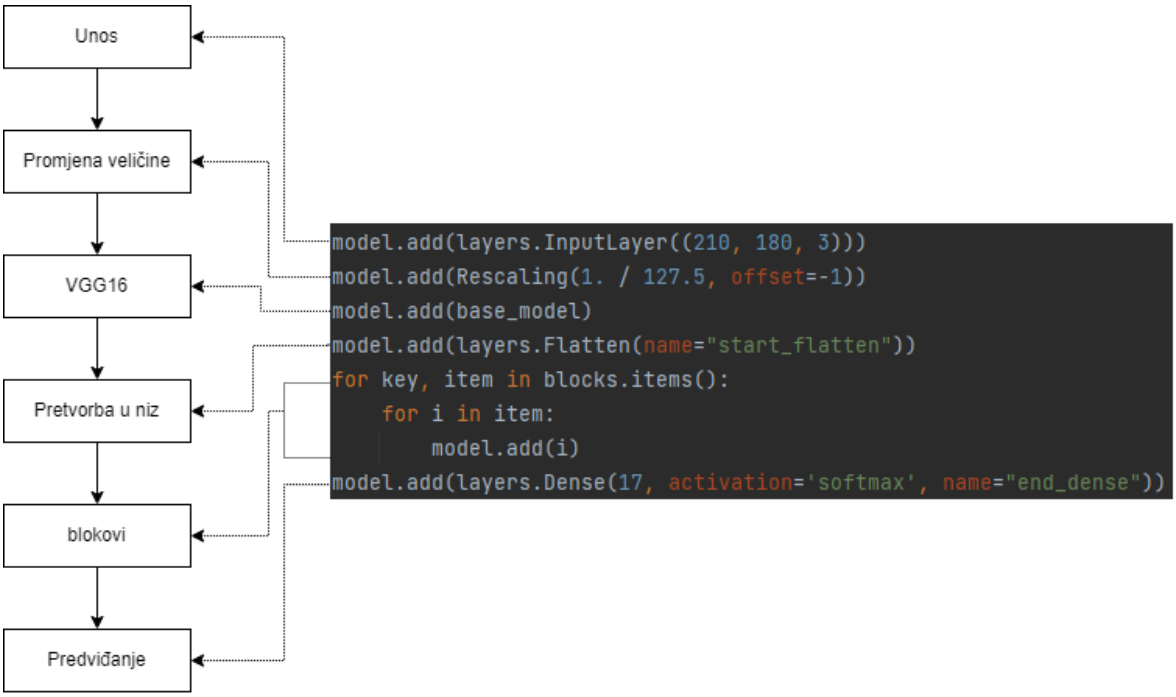
Preciznost modela ukazuje na preciznost predviđanja sadržaja slike, odnosno omjer točnih i krivih predviđanja kategorije objekta na slici. Gubitak modela ukazuje na to koliko dobro model opisuje skup ulaznih podataka, ako govorimo o gubitku za treniranja, odnosno koliko je dobra izvedba modela (engl. *model performance*) kada govorimo o gubitku validacije.

Kao osnovu za odabir strukture bloka odabrana je ista veličina gustih slojeva u istom bloku, te sloj za izbacivanje (engl. *dropout*).

Kako bi model (Slika 22) ostvario bolju preciznost bilo je potrebno usporediti performance algoritama za optimizaciju neuronskih mreža, često se nazivaju samo optimizatori, kako bi se pronašao optimalni algoritam za skup ulaznih podataka. Kako bi usporedba performansi bila vjerodostojna, broj blokova te struktura samog bloka mora ostati nepromijenjena za sve optimizacije, kao i broj epoha (engl. *epoch*). Najčešće korišteni optimizatori dostupni u TensorFlow biblioteci su Adam i SGD, oba algoritma su dobra te imaju i prednosti i mane (Tablica 3). Za izradu modela korišteni su sljedeći slojevi:

- *InputLayer*: Sloj za unos, ulazna točka mreže, definira format ulaznog podatka
- *Rescaling*: Sloj za promjenu ulaznih veličina, ulazne vrijednosti se množe s parametrom scale, te im se dodaje parametar offset
- *Flatten*: Sloj pretvara ulaz u jednodimenzionalni sloj
- *Dense*: Potpuno povezani sloj, parametar activation definira aktivacijsku funkciju koja će biti korištena
- *Dropout*: Sloj za nasumično postavljanje ulaznih vrijednosti na 0

Potpuno povezani sloj je korišten s dvije različite aktivacijske funkcije Softmax i ReLu. Softmax aktivacijska funkcija korišten je za sloj za predviđanje, korištenjem ove funkcije izlazi modela se transformiraju niz vjerojatnosti. ReLu aktivacijska funkcija korištena je u potpuno povezanim slojevima blokova. Ova funkcija za vrijednost vraća ulaznu vrijednost samo ako je ona pozitivna u protivnom rezultat je 0.



Slika 22: Struktura modela i vizualizacija toka

Tablica 3: Usporedba Adam i SGD optimizatora

	Adam	**SGD**
Vrijeme treniranja	4h 25min	7h 15min
Broj epoha	30	30
Preciznost	59,01%	70,03%
Gubitak	1,4822	0,9212

Za treniranje modela na Google Colaboratory platformi, optimizator Adam se pokazao odličan za treniranje modela kako bi se ocijenili promijenjeni parametri, iako je davao nižu preciznost od modela sa SGD optimizatora, povećanje preciznost u modelima s Adam optimizatorom je ukazivalo da će preciznost modela sa SGD optimizatorom također rasti, ali teško je bilo odrediti koliki točno utjecaj će imati na krajnju preciznost modela sa SGD optimizatorom. Znatan dio treniranja odrađen je korištenjem Adam optimizatora, kako bi se uštedjelo na vremenu, što je dovelo do potrebe za dodatnim promjenama u trenutku prelaska na SGD.

Korištenjem Adam optimizatora, najveća postignuta preciznost, nakon testiranja s raznim parametrima, bila je 67%, što nije bilo zadovoljavajuće. Zahvaljujući znatno kraćem vremenu potrebnom za treniranje modela, uzimajući u obzir sve promjenjive parametre te kako su utjecali na promjenu preciznosti i dužinu treniranja, prelazak na SGD optimizator i pronalaženje optimalnih parametara, bilo je znatno lakše.

Rezultati dobiveni s inicijalnim postavkama za SGD optimizator znatno su bile bolje nego rezultati s Adam optimizatorom. Iako preciznost još nije bila zadovoljavajuća, u samo nekoliko promjena postignuta je preciznost od 83,28% koja je u ovom slučaju zadovoljavajuća.

U nastavku u tablično prikazu nalaze se 2 modela, odnosno pristupa izradi modela ovisno o redoslijedu blokova koji se koriste, i njihovim parametrima. Prvi pristup koristi uzlazne blokove, odnosno svaki sljedeći blok ima veći ili jednak broj neurona od prethodnog. Iako je nekonvencionalan, ova metoda, uz dovoljno velik broj podataka, bi se mogla pokazati uspješnom, ako se koristi već gotovi model. Zbog relativno malog broja podataka, rezultati nisu bili zadovoljavajući. Druga metoda koristi jednak ili manji broj neurona u svakom sljedećem bloku. Svaka struktura blokova trenirana je najmanje tri puta, kako bi se dobili podaci o preciznosti te kako bi se lakše ocijenila performansa modela. Skup podataka za treniranje pripremljen je za rješavanje klasifikacijske razine 2 te su svi modeli trenirani na istom skupu.

Parametri modela korištene prilikom procesa treniranja su funkcija za rano zaustavljanje "*EarlyStopping*", u nastavku es, funkcija za prilagodbu brzine učenja "*LearningRateOnPlateau*", u nastavku lr, "*sparse_categorical_crossentropy*" za funkciju gubitka te "accuracy" za metriku modela.

Funkcija es je funkcija koja se koristi za zaustavljanje procesa treniranja modela, promatrajući vrijednost određene metrike. Ova funkcija se koristi kako bi se spriječilo pretreniranje modela, što kao rezultat ima smanjenje vremena treniranja. Funkcija lr je funkcija koja se koristi za prilagodbu hiper-parametra brzine učenja. Ova funkcija se koristi kako bi se prilikom treniranja modela, u odnosu na promatranu vrijednost mijenjao hiper–parametar brzine učenja. Na slici 25 prikazana je inicijalizacija ovih funkcija.

```
es = Es(
  monitor='val_accuracy',
  mode='max',
  patience=5,
  restore_best_weights=True
)
rl = Lr(
    monitor='val_accuracy',
    patience=3,
    min_lr=0.000001
)
```

Ispis 3: Inicijalizacija es i lr objekata

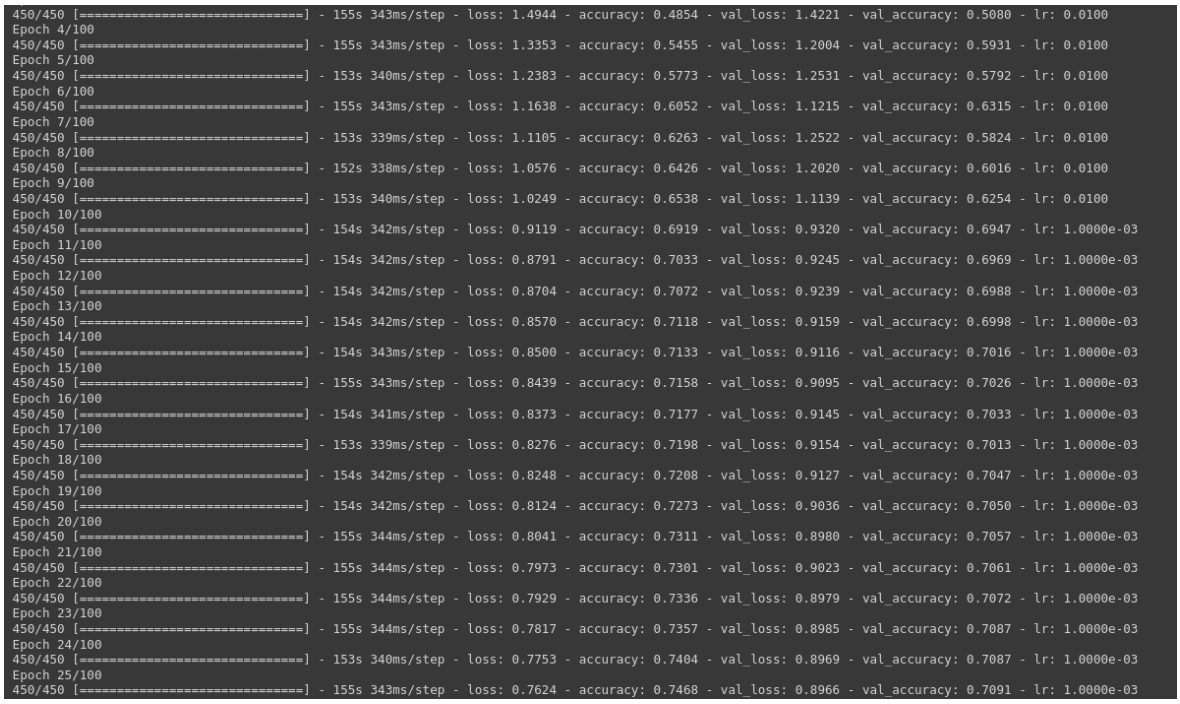
Prilikom treniranja modela, odabrana metrika koju nadziru es i lr je validacijska preciznost. Ova metrika je odabrana kao konstanta nakon testiranja kombinacija validacijske preciznosti i gubitka za obje metrike, kao i nadziranje obje istovremeno, jer je pokazala najbolji rezultat. Za lr najmanja dozvoljena brzina učenja je također postavljena kao konstanta, na vrijednost 1×10^{-6} korištenjem parametra "*min_lr*". Vrijednost koja je mijenjana prilikom treniranja je "strpljenje" odnosno maksimalni broj epoha bez poboljšanja vrijednosti koja se nadzire, a korištena je zadana vrijednost 0,1, za faktor mijenjanja brzine učenja.

Za funkciju gubitka odabrana je "*sparse_categorical_crossentropy*", ova funkcija se razlikuje od "*categorical_crossentropy*" jedino po načinu definiranja ulaznih parametara. Za "*categorical_crossentropy*" kategorije moraju biti predstavljene pomoću "*categorical vector*", na primjer [1,0,0], [0,1,0] i [0,0,1] za kategorije 1, 2 i 3. Kod "*sparse_categorical_crossentropy*" nazivi kategorija su cjelobrojne vrijednosti, na primjer [1], [2], [3] za kategorije 1,2 i 3.

Treniranje modela (Ispis 4) započinje se pozivom funkcije fit. Parametri proslijeđeni ovoj funkciji su skup podataka za treniranje, skup podataka za validaciju, veličina skupine, broj epoha odnosno ciklusa i lista funkcija koja se izvršava na kraju svake epohe. Ispis procesa treniranja prikazan je na slici 27.

```
model.fit(
  train_ds,
  epochs=epochs,
  validation_data=valid_ds,
  batch_size=32,
  callbacks=[es, lr]
)
```

Ispis 4: Poziv funkcije fit



Slika 23: Proces treniranja

U tablicama koristi se nekoliko simbola, d(AxB) koji predstavlja gusti blok, A broj gustih slojeva, a B broj neurona gustog sloja, do(X) predstavlja sloj smanjenja, a X vrijednost smanjenja, broj između 0 i 1. U "*callbacks*", nalaze se sve funkcije dodatne funkcije korištene za treniranje modela, tako na primjer vrijednost "es - 5" označava funkciju es koja ima parametar strpljivosti 5 epoha. U stupcu optimizatora, broj predstavlja granularnost po kojoj se koristi optimizator, niža vrijednost u konačnici znači veća preciznost, ali vrijeme treniranja se znatno poveća. Sve modele nije bilo moguće trenirati do kraja, takvi modeli nalaze se u posebnoj tablici (Tablica 6). Razlog neuspješnog treniranja je vremensko ograničenje.

Tablica 4: Modeli s uzlaznim blokovima

struktura blokova	**optimizator**	**veličina grupe**	**callbacks**	**gubitak**	**preciznost**
d(2x4096) do(0.4) d(2x8192) do(0.5)	Adam - 0.0001	32	es - 5	2,50 - 2,29	58% - 59,69%
d(2x1024) do(0.3) d(2x2048) do(0.25) d(2x4096) do(0.3) d(2x8192) do(0.5)	Adam - 0.0001	32	es - 5	2,22 - 2,35	62,08% - 66,95%
d(2x2048) do(0.2) d(3x2048) do(0.4) d(2x4096) do(0.1) d(2x8192) do(0.3) d(1x8192) do(0.25)	SGD - 0.0001	32	es - 5 lr - 3	2,1 - 2,5	63,5% - 69%
d(2x4096) do(0.25) d(2x8192) do(0.35) d(2x16834) do(0.25)	SGD - 0.0005	32	es - 5 lr - 3	1,1 - 1,35	63% - 72%
d(3x8192) do(0.35) d(3x16834) do(0.25)	SGD - 0.0001	32	es - 5 lr - 3	2,02 - 2,24	60% - 73%

Tablica 5: Modeli sa silaznim blokovima

struktura blokova	**optimizator**	**veličina grupe**	**callbacks**	**gubitak**	**preciznost**
d(2x8192) do(0.25) d(2x4096) do(0.35)	Adam - 0.0001	32	es - 5	2,1 - 2,29	62% - 63,72%
d(2x8192) do(0.25) d(2x4096) do(0.35) d(2x2048) do(0.25) d(2x1024) do(0.35)	Adam - 0.0001	32	es -5	2,22 - 2,35	62,5% - 65,1%
d(1x8192) do(0.25) d(2x8192) do(0.35) d(2x4096) do(0.25) d(3x2048) do(0.35) d(2x2048) do(0.25)	SGD - 0.0001	32	es - 5 lr - 3	1,1 - 2,9	66% - 72,82%
d(2x16834) do(0.2) d(2x8192) do(0.3) d(2x4096) do(0.2)	SGD - 0.0005	32	es - 5 lr - 3	1,1 - 1,35	77% - 78,74%
d(3x16834) do(0.3) d(3x8192) do(0.2)	SGD - 0.0001	32	es - 5 lr - 3	2,02 - 2,24	80% - 81,2%
d(2x4096) do(0.3)	SGD - 0.0001	64	es - 5 lr - 3	1,7 - 4,04	80,15% - 83,28%

struktura blokova	optimizator	veličina grupe	callbacks	gubitak	preciznost
d(2x4096) do(0.2)					

Tablica 6: Modeli koji nisu uspješno trenirani

struktura blokova	optimizator	veličina grupe	callbacks
d(3x8192) do(0.5) d(3x4096) do(0.5)	SGD / ADAM	16 / 32 / 64	es(5) lr(3)
d(2x4096) do(0.5) d(2x8192) do(0.5)	SGD / ADAM	64	es(5) lr(3)
d(2x2048) do(0.75) d(2x2048) do(0.75) d(2x2048) do(0.5)	SGD / ADAM	16 / 32 / 64	-
d(2x2048) do(0.75) d(2x2048) do(0.75) d(2x2048) do(0.5)	SGD / ADAM	16	es(5) lr(3)
d(2x2048) d(2x1024) do(0.08) d(2x1024) d(2x512) do(0.1) d(3x512) do(0.1) d(4x256) d(0.1)	SGD / ADAM	16 / 32 / 64	-
d(2x2048) d(2x1024)	SGD	16	es(5) lr(3)

struktura blokova	optimizator	veličina grupe	callbacks
do(0.08) d(2x1024) d(2x512) do(0.25) d(3x512) do(0.25) d(4x256) do(0.25)			
d(2x1024) do(0.1) d(2x1024) do(0.1) d(2x1024) do(0.2) d(2x512) do(0.1) d(2x512) do(0.1) d(2x512) do(0.2) d(4x256) do(0.2) d(4x128) do(0.2)	SGD / ADAM	16 / 32 / 64	-

U tablicama Tablica 4 i Tablica 5 prikazani su rezultati nakon odabira najboljih vrijednosti za es, lr i brzinu učenja optimizatora ADAM ili SGD koji daju vrijeme treniranja manje od 8 sati. Uzlazni blokovi iako imaju manju preciznost uspješno su trenirani na jednoj kategoriji više nego silazni, a to je kategorija "japanke", iako ljudskom oku su znatno različite od kategorije "šlape", prilikom treniranja utvrđeno je da za računalo ove dvije kategorije su izuzetno teške za prepoznati i naučiti njihove karakteristike. Stoga u silaznim modelima ove kategorija "japanke" je dodana u kategoriju "šlape" kako bi se povećala preciznost modela. U Tablici 6 prikazane su neke od struktura modela koji nisu uspješno trenirani. Glavni razlog za bezuspješno treniranje je vrijeme trajanja treniranja te u rijetkim slučajevima radna memorija računala je bila nedostatna. U ovoj tablici svi modeli koji dijele strukturu i "*callbacks*" tako imaju samo jedan unos u tablici, na primjer prvi redak tablice sadržava model koji za vrijednost optimizatora ima "SGD / ADAM" ovakav unos označava

da je model treniran korištenjem ADAM i SGD optimizatora, također isti način označavanja se koristi u stupcu “veličina grupe”.

Procesi treniranja i testiranje, odnosno uklapanja (engl. *fitting*), odviju se jedan za drugim u više ciklusa, odnosno epoha. Po završetku jedne epohe, ocjenjuje se performansa modela pomoću funkcije gubitka i pogreške te se vrši korekcija težina i pristranosti.

Nakon treniranja modela sa zadovoljavajućom preciznosti, uslijedilo je završno treniranje modela. Odabrani model treniran je još četiri puta, jedan put za svaku klasifikacijsku razinu. Jedina razlika u procesu treniranja bila je struktura podataka za treniranje. Klasifikacijska razina 1 ima samo dvije kategorije 0 i 1, gdje kategorija 0 označava žensku odjeću i obuću, a kategorija 1 mušku. Iako ovaj problem spada u binarnu klasifikaciju, koja u pravilu daje najveće preciznosti, zbog raznovrsnosti podataka te sličnosti u određenim predmetima ova razina je imala najgoru preciznost.

Treniranje klasifikacijske razine 2, odnosio se na “grubu” klasifikaciju odjeće i obuće. Klasifikacija obuće sadržavala je kategorije košulje, majice, dok za obuću među kategorijama nalazile su se čizme i cipele. Ova kategorija je ujedno i najveća po veličini klasifikacijskih opcija, ukupno 17 kategorija klasifikacije.

Klasifikacijska razina 3 sadržana je od preciznije klasifikacije, tako na primjer kategorija čizme u ovoj razini se dijelila na niske čizme, visoke čizme, visoka potpetica, debela potpetica i slične kategorije. Ova razine nije definirana za sve kategorije iz razine 2, odnosno samo 12 kategorija sadrži potkategorije. Čizme spadaju u razinu 2 s najviše potkategorija, ukupno 6, dok su neke kategorije imale samo dvije potkategorije.

Klasifikacijska razina 4 najpreciznije definira odjeću ili obuću. Ova razina ujedno je i najmanje zastupljena razina. Kao i razina 3 nije obavezna. Kao primjer ove razine može se utezi kategorija niske čizme. Potkategorije ove kategorije razine 3 su radne, za snijeg, luksuzne. Ova klasifikacijska razina definirana je samo u 4 slučaja, od kojih sva 4 spadaju u obuću.

Struktura programskog rješenja kao ulazni parametar prima jednu sliku. Model klasifikacijske razine 1 koristi sliku kao ulaz te daje vrijednost u skupu [0,1]. Razina 2 zatim prima istu sliku te vrši predviđanje što se nalazi na slici te kao rezultat daje broj iz skupa [0, 17]. Na temelje izlaza razine 2, provjerava se postoje li razina 3, odnosno postojanje modela 3.X, gdje X predstavlja izlaznu vrijednost razine 2. Kada razina 3 nije definirana proces je

završen, dok ako razina 3 je, model za razinu 3 se učitava te za ulaz prima sliku i vrši predviđanje kategorije razine 3, skup vrijednosti ove razine je promjenjiv ovisno o broju kategorije. Zatim se provjerava postoji li model 4.X.Y gdje X označava izlaz razine 2, a Y izlaz razine 3. U slučaju da model ne postoji proces se završava, a u suprotnom slika se prosljeđuje modelu 4.X.Y.

Rezultat procesa je niz od 4 broja gdje pozicija broja u nizu odgovara klasifikacijskoj razini. Tako iz rezultata [1, 6, -1, -1] se može zaključiti da ulazna slika pripada kategoriji muško u razini 1, a kategoriji košulje u razini 2 te da razine 3 i 4 ne postoje.

U nastavku se nalazi tablični prikaz za modele trenirane na najvećem skupu podataka. Poredani po klasifikacijskim razinama, svakom modelu pridijeljen je opis, odnosno kategorija koju model pobliže definira.

Ukupnu preciznost ovakvog sustava teško je odrediti, točnije izračunati. Razine 1 i 2 su uvijek prisutne i prosječnu preciznost ove dvije razine je relativno lako izračunati. Kada se u takvu računicu dodaju razine 3 i 4 koje nisu jednako zastupljene u skupu podataka za treniranje te ne moraju biti definirane, sami izračun postaje znatno kompliciraniji. Iako razine 3 i 4 bi znatno manje utjecale na ukupnu preciznost sustava od razine 1 i 2.

Tablica 6: Trenirani modeli i njihove preciznosti

Razina	Model	Opis	Preciznost
1	model.1	spol	63,15%
2	model.2	“grube” kategorije	83,98%
3	model.3.2	cipele	81,13%
3	model.3.3	majice	80,92%
3	model.3.4	čizme	82,11%
3	model.3.6	košulje	82,50%
3	model.3.9	hlače	80,01%
4	model.4.2.2	tenisice	78,73%
4	model.4.2.9	cipele	71,57%
4	model.4.3.1	polu-formalne majice	72,25%
4	model.4.4.2	radne čizme	68,90%

Prilikom izrade modela, naišao sam na nekoliko problema. Među kojima su pogrešna kategorizacija slika, sličnost različitih kategorija koje model nije mogao razlikovati. Pogrešna klasifikacija je jedan od najčešćih problema pri izradi modela za prepoznavanje objekata u slici. Poželjno je da 100% skupa za treniranje bude dobro numerirano, kako se ovaj proces odvijao ručno, mogućnost pogreške je velika. Ovakva pogreška može za posljedicu imati znatno smanjenje preciznosti modela, ali moguće ju je otkloniti, što je i napravljeno nakon što je utvrđen problem. Drugi problem na koji sam naišao je sličnost kategorija, sličnost kategorije "japanke" i kategorije "šlape", iako ljudskom oku prepoznavanje ovih dviju kategorija je relativno lako, kada su slike ovakvih proizvoda slikane iz određenih pogleda, odnosno kutova, čak ni ljudsko oko ne može jednostavno napraviti razliku. Kao najteži primjer za razlikovanje ovih kategorija je kada slika prikazuje samo đon, tada ove dvije kategorije se ne mogu jednostavno razlikovati. Ovakve slike nisu korisne i u pravilu se ne nalaze u skupovima za treniranje modela. U nastavku je primjer korištenih slika (Slika 24) koje su uzrokovale problem. Ovaj problem riješen je spajanjem kategorija "šlape" i "japanke" u jednu kategoriju. Ovakvo rješenje dalo je znatno bolju preciznost uz uklanjanje dvije kategorije i stvaranje nove.

Slika 24: Slika iz kategorije "šlape" (lijevo) i kategorije "japanke" (desno)

7 ZAKLJUČAK

Odabir i optimizacija modela pokazala se izazovnom, zbog ograničenja u resursima dostupnima za treniranje.

Glavna zapreka prilikom izrade modela, osim relativno malog broja slika, bilo je ograničenje u vremenu treniranja, a zatim u korištenoj memoriji računala. Google Colaboratory platforma podržava najdulje vrijeme kontinuiranog treniranja od osam sati. Također postoje i varijabilni limiti ukupno iskorištenih resursa za tjedan i mjesec dana, što se pokazao u konačnici kao najveći problem. Nakon kontinuiranog korištenja platforme 4 do 5 dana, tjedni limit bi se aktivirao i trebalo je čekati novi tjedan kako bi se nastavilo s testiranjem i treniranjem modela. Drugo najveće ograničenje je količina pohrane unutar Google Colaboratory platforme. Dobiveni prostor za pohranu podataka može biti različit, a prilikom izrade ovog rada najmanja zabilježena dobivena vrijednost prostora za pohranu je bila 13 GiB (engl. *GibiByte*). Uzimajući u obzir ukupnu veličinu podataka za treniranje i testiranje te veličinu modela od 2,3 do 5,8 GiB, prostor za pohranu podataka je bio "jedva" dostatan.

Ako izuzmemo ograničenja resursa, najviše vremena potrošeno je na odabir i podešavanje dodatnih funkcija modela. Kada se više ovakvih funkcija koristi zajedno, iako imaju minimalni utjecaj jedna na drugu, svaka od njih znatno utječe na preciznost modela. Što je nekada davalo privid da utječu jedna na drugu. Najveći problem podešavanja parametara ovih funkcija je što je rezultat vidljiv tek kada je model treniran do kraja. Tako za najmanje promjene ovih parametara, provjera poboljšanja je trajala i do osam sati.

U konačnici isti rezultati, s istim postupcima, mogu se postići u znatno manjem vremenu, čak i na slabijim računalima, ako se računala koriste cijeli dan, odnosno ako ne postoje limitacije koje daje Google Colaboratory platforma.

Poboljšanja modela su moguća, povećanjem skupa za treniranje, sporijim treniranjem, odnosno smanjenje brzine učenja modela, primjenom veće neuronske mreže ili promjenom svih navedenih stavki. Popratno "cijena" treniranja modela bi se znatno povećala. Neke od mogućih "poskupljenja" su zahtjev za većom količinom prostora za pohranu podataka, veća količina radne memorije potrebna za treniranje modela i duže vrijeme treniranja. Primjenom svih navedenih postupaka moguće je znatno poboljšanje modela, međutim "cijena" treniranja bi se također znatno povećala.

Za poboljšanje cjelokupnog sustava, prvi korak bi bio prolazak procesa izrade modela, koji je opisan u poglavlju Izrada modela, za svaku klasifikacijsku razinu posebno, odnosno za svaki krajnji model.

○ LITERATURA

[1] M. Kljaković Gašpić, “Računalni vid u primjeni,” Sveučilišni odjel za stručne studije, Sveučilište u Splitu, 2020.

[2] M. Hrga, “Računalni vid,” Zbornik radova Veleučilišta u Šibeniku, vol. 1–2, pp. 207–216, 2018.

[3] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed. Upper Saddle River, NJ: Pearson, 2017.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. London, England: MIT Press, 2016.

[5] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed. Cham, Switzerland: Springer Nature, 2023.

[6] J. E. Solem, Programming computer vision with python: Techniques and libraries for imaging and retrieving information, 1st ed. Sebastopol, CA: O’Reilly Media, 2012.

[7] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. Cambridge, England: Cambridge University Press, 2011.

[8] F. Chollet, Deep learning with python. New York, NY: Manning Publications, 2017.

[9] N. Bolf, “Osvježimo znanje: Strojno učenje,” Kemija u industriji: Časopis kemičara i kemijskih inženjera Hrvatske, vol. 70, no. 9–10, pp. 591–593, 2021

[10] E. Alpaydin, Introduction to Machine Learning, 4th ed. London, England: MIT Press, 2020.

[11] O. Theobald, Machine learning for absolute beginners: A plain English introduction (third edition). Scatterplot Press, 2021.

[12] P. Harrington, Machine Learning in Action. Manning Publications, 2012.

[13] A. Geron, Hands-on machine learning with scikit-learn, keras, and tensorflow, 3rd ed. O’Reilly Media, 2022.

[14] H. Kumar, “Data augmentation techniques,” OpenGenus IQ: Computing Expertise & Legacy, 15-Apr-2019. [Online]. Dostupno: https://iq.opengenus.org/data-augmentation/. [Posjećeno 02.05.2023].

[15] J. Grus, Data science from scratch: First principles with python, 2nd ed. Sebastopol, CA: O’Reilly Media, 2019.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” Nature, vol. 323, no. 6088, pp. 533–536, 1986.

[17] R. E. Bellman, Dynamic programming. Princeton, NJ: Princeton University Press, 1957.

[18] G. Blokdyk, Tensorflow: A Complete Guide. 5starcooks, 2018.

[19] Y. E. Wang, G.-Y. Wei, and D. Brooks, “Benchmarking TPU, GPU, and CPU platforms for deep learning,” arXiv [cs.LG], 2019. [Posjećeno: 20.12.2022].

[20] M. ul Hassan, “VGG16 - convolutional network for classification and detection,” Neurohive.io, 20-Nov-2018. [Online]. Dostupno: https://neurohive.io/en/popular-networks/vgg16/. [Posjećeno: 20.12.2022].