Patrik Črnčec, univ. bacc. inf.[1]
Doc. dr. sc. Darko Andročec[1]

PRIMJENA STROJNOG UČENJA NA PROBLEMU KOLORIZACIJE CRNO-BIJELIH SLIKA

Stručni rad / Professional paper
UDK 004.85.004.421

Ovaj rad se bavi strojnim učenjem i njegovim korištenjem u svrhu izrade aplikacije za kolorizaciju crno-bijelih slika. Kao pomoć pri izradi aplikacije su se koristile otvorene biblioteke za strojno učenje. U prvom dijelu rada je objašnjeno strojno učenje, njegova podjela i njegovi glavni algoritmi. Također su objašnjene i međusobno uspoređene neke od najpopularnijih otvorenih biblioteka za strojno učenje. U drugom dijelu je objašnjen rad izrađene aplikacije, te algoritma strojnog učenja koji se je koristio pri njezinoj izradi. Za kraj su prikazani rezultati aplikacije zajedno s evaluacijom implementiranog modela.

Ključne riječi: *strojno učenje, biblioteke, crno-bijele slike, kolorizacija, autoenkoder.*

1. Uvod

Strojno učenje je postalo u posljednjih nekoliko godina veoma popularno zbog svojih brojnih primjena. Rasprostranjena primjena metoda strojnog učenja je moguća zahvaljujući sve boljim računalnim performansama, te sve većoj količini podataka koje je moguće upotrijebiti za treniranje modela strojnog učenja. Kako su često modeli strojnog učenja složeni, razvijene su otvorene biblioteke koje znatno olakšavaju i ubrzavaju implementaciju algoritama strojnog učenja u željene aplikacije.

Cilj ovog rada je objasniti strojno učenje i njegove glavne algoritme te navesti i opisati otvorene biblioteke za strojno učenje koje se danas najčešće koriste. Osim toga, cilj je uz pomoć otvorenih biblioteka za strojno učenje izraditi aplikaciju za kolorizaciju crno-bijelih slika. Ideja ove aplikacije je proizašla iz uočavanja da su prethodne metode za kolorizaciju crno-bijelih slika bile vremenski neefikasne, te su zahtijevale stručno znanje u obradi slika. Jedna od takvih neefikasnih metodi je bojanje slika u popularnom alatu Adobe Photoshop, gdje čovjek kolorizira sliku na temelju njegove vlastite percepcije i mišljenja kako bi kolorizirana verzija te slike mogla izgledati. Za razliku od toga, algoritam strojnog učenja je u stanju s određenom mjerom preciznosti naučiti preslikavanje crno-bijelih piksela u RGB piksele te tako automatizirati proces kolorizacije.

Ovaj se stručni rad sastoji od pet poglavlja, gdje se u drugom i trećem poglavlju opisuje teorijska podloga strojnog učenja i njezinih otvorenih biblioteka, dok se u četvrtom poglav-

[1] Fakultet organizacije i informatike, Sveučilište u Zagrebu

lju opisuje rad izrađene aplikacije te donose njezini rezultati. Rad završava sa petim poglavljem u kojem se izvodi konačni zaključak.

2. Strojno učenje

Strojno učenje područje je umjetne inteligencije koje pruža sustavima svojstvo da uče kroz iskustvo bez da su eksplicitno programirani. Takvi sustavi koriste velike količine podataka kako bi kroz proces učenja naučili u njima prepoznati uzorke koje će utjecati na donošenje odluka u budućnosti (Team, 2020).

Strojno učenje omogućuje rješavanje problema koji su prije njegove pojave bili neriješivi. Primjerice, prije nije bilo moguće napraviti program koji bi uspješno mogao raspoznati da li se na nekoj slici nalazi pas ili ne, te ako se nalazi, u kojem se to području slike nalazi. Pisanje koda koji bi provjeravao boju svakog piksela i na temelju toga donio odluku bi bilo besmisleno i nemoguće jer se sigurno ne može pokriti sve slučajeve kao što su različite pasmine, različite pozadine (npr. snijeg, livada, unutrašnjost kuće…), kut slikanja, rezolucije slika, crnobijele slike, osvjetljenje… Strojno učenje ovaj problem može riješiti tako što će algoritam uz pomoć odabranih podataka odnosno slika kroz vrijeme naučiti koje sve elemente slika mora sadržavati da bi se na njoj potencijalno mogao nalaziti pas. Neki od elemenata mogu biti primjerice pasje uši, rep, oblik tijela, proporcije tijela i slično.

2.1. Područja primjene strojnog učenja

Prethodni primjer je samo jedan od mnogobrojnih primjena strojnog učenja u stvarnom svijetu. Strojno učenje se danas upotrebljava u gotovo svim industrijama. U nastavku će biti nabrojena neka od područja njegove primjene te će se za pojedino područje objasniti na koji se način primjenjuje.

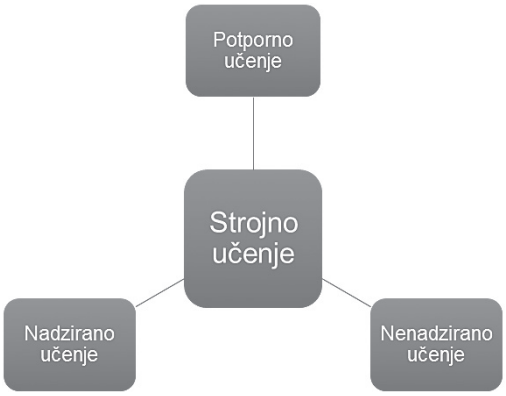
Trgovine mogu koristiti strojno učenje za analizu njihovih podataka o prodaji kako bi saznali ponašanje njihovih kupaca te tako poboljšali njihov odnos. Osim toga, moguća je i upotreba chatbotova kao pomoć kupcu pri kupovini proizvoda kroz pružanje informacija o trgovini i njezinim proizvodima ili uslugama. Financijske institucije analiziraju prošle transakcije kako bi predvidjele kreditne rizike klijenata i tako odbile ili prihvatile novi zahtjev za kredit klijenta. Roboti uče optimizirati svoje ponašanje kako bi mogli završiti neki zadatak koristeći što manje resursa. Također, roboti mogu koristiti tehnike strojnog učenja kako bi se mogli snaći u prostoru. U bioinformatici se strojno učenje koristi za analizu bioloških podataka. U automobilskoj industriji strojno učenje se koristi za implementaciju autonomnih vozila. Digitalni asistenti mogu prepoznati pitanje koje je postavio čovjek te na to pitanje dati smislen odgovor ili izvršiti odgovarajuću akciju (Alpaydin, 2010).

2.1. Podjela strojnog učenja

Strojno učenje nije samo jedan algoritam koji omogućava prethodno navedene funkcionalnosti, već je to jedna velika skupina algoritama koja se dijeli na tri glavna područja. Svaki od tih algoritama ima svoju svrhu, a odabir algoritma, tj. metode koja će se koristiti pri

implementiranju rješenja ovisi ponajviše o zadanom problemu. Prema tome, strojno učenje se dijeli na: nadzirano učenje, nenadzirano učenje i podržano učenje.

Slika 1. Podjela strojnog učenja



Izvor: (Nasteski, 2017)

Glavna zadaća algoritama nadziranog učenja je generiranje funkcije koja preslikava ulaze na željene izlaze u svrhu predviđanja (regresija) ili klasificiranja podataka. Izlazni podaci iz klasifikacijskih modela su diskretne vrijednosti dok su kod prediktivnih kontinuirane. Kod klasifikacijskog modela postoje klase koje se moraju odrediti prije procesa treniranja zbog toga što su one ujedno i izlaz na temelju kojeg algoritam uči. Klase su konačan skup podataka koje definira čovjek na način da svaka klasa sadrži one podatke koji su međusobno slični ili identični po jednom ili više obilježja. Kod prediktivnih modela se uz pomoć obilježja ulaznog podatka predviđa izlazna vrijednost, konkretnije neki realan broj (Nasteski, 2017).

Proces učenja kod nadziranog strojnog učenja se dijeli na dvije faze: treniranje i testiranje. Kako bi to bilo moguće, potrebno je skup podataka podijeliti u dvije zasebne cjeline, u podatke za treniranje i testiranje, gdje će najveći udio imati podaci za treniranje. U fazi treniranja se podaci za treniranje uzimaju kao ulazni, te se njihove karakteristike uče uz pomoć odgovarajućeg algoritma. Glavni zadatak algoritma je predvidjeti izlazni podatak na temelju karakteristika ulaznog podatka, kako bi onda svoju predviđenu vrijednost mogao usporediti sa stvarnim izlaznim podatkom i pritom otkriti greške. Nakon toga se na temelju pronađene greške model modificira kako bi buduće predikcije mogle biti preciznije, što zapravo daje svojstvo učenja. U fazi testiranja se uzima istrenirani model te se uz pomoć njega rade predikcije nad podacima za testiranje. S obzirom da podaci za testiranje nisu bili korišteni prilikom treniranja modela, rezultat predikcije nad njima će poslužiti za evaluaciju, odnosno dobiva se informacija o tome da li je istrenirani model dovoljno dobar za naše potrebe ili ga je potrebno dodatno usavršiti (Nasteski, 2017).

Primjer predviđanja uz pomoć nadziranog učenja je predviđanje cijene kuće na temelju njezine lokacije, veličine, starosti (godine izgradnje), interijera i slično. Primjer klasifikacije može biti prepoznavanje i klasifikacija rukom napisanih brojeva gdje sustav na temelju slike određuje koji se broj nalazi na slici. U ovom su slučaju klase znamenke u dekadskom brojev-

nom sustavu (0-9). Također jedan primjer može biti i klasifikacija da li je mrežni promet maliciozan u sustavima Interneta stvari (Oreški and Andročec, 2018).

Za razliku od nadziranog učenja gdje je cilj naučiti preslikavanje iz ulaznog u izlazni podatak, u nenadziranom učenju nemamo izlazne podatke, već imamo ulazne podatke uz pomoć kojih želimo pronaći pravilnosti. U ovom je području čest pojam procjena gustoće koji predstavlja pronalaženje uzoraka u prostoru ulaznih podataka koji se pojavljuju češće nego neki drugi uzorci. Jedna od najčešćih metoda za procjenu gustoće je klasterifikacija odnosno grupiranje ulaznih podataka (Alpaydin, 2010).

Podržano učenje se bavi problemom pronalaženja i odabira odgovarajućih akcija koje je potrebno poduzeti kako bi se maksimizirala nagrada. Za razliku od nadziranog i nenadziranog učenja, ovdje nemamo podatke, već ih je potrebno saznati iz postupaka pokušaja i pogreške. U podržanom učenju najčešće imamo niz stanja i akcija koje su u interakciji s okolinom. Često trenutna akcija utječe na nagradu u trenutnom vremenu i u nadolazećem vremenu, tj. ona može utjecati na odluku daljnjih akcija. Glavno obilježje podržanog učenja je kompromis između istraživanja i eksploatacije. U istraživanju sustav isprobava nove akcije da utvrdi koliko su one efikasne, dok kod eksploatacije sustav koristi već ranije poznate akcije koje daju visoku nagradu (Bishop, 2006).

Podržano učenje je okvir opće namjene za kreiranje inteligentnih agenata. Ukoliko agentu pružimo okolinu i nagradu, on će naučiti djelovati s tom okolinom u cilju da maksimizira svoju nagradu. Ovaj princip učenja sliči onom kod živih bića. Koristi se u raznim područjima kao što su: autonomna vozila, robotika, računalne igre, optimizacija pozicioniranja oglasa, trgovanje dionicama (Buduma and Locascio, 2017)… Na primjer, kod autonomnih vozila okolina je promet, a cilj agenta (automobila) je da vozi po prometnoj traci kako bi na taj način maksimizirao svoju nagradu. Osim toga, mora izbjeći sudare jer oni donose negativnu nagradu odnosno kaznu.

3. Otvorene biblioteke za strojno učenje

Biblioteke su kolekcija resursa koje koriste računalni programi u svrhu razvoja programa. Resursi mogu biti: konfiguracije, dokumentacija, podaci za pomoć, programski kod i slično. Osim toga, biblioteka je kolekcija implementacije ponašanja koja je napisana u nekom programskom jeziku s dobro definiranim sučeljem za pristup odnosno prizivanje ponašanja. Njezin glavni element je unaprijed napisani programski kod koji je organiziran tako da se može koristiti na više različitih programa ili podprograma koji ne ovise jedan o drugom. Još jedno važno svojstvo je da za korištenje neke biblioteke, korisnik ne mora znati interne detalje biblioteke već samo njezino sučelje za pristup. Kada program pozove neku biblioteku on dobiva ponašanje implementacije unutar nje bez potrebe zasebne implementacije za ostvarenje tog ponašanja, pa iz toga onda proizlazi njezina svrha korištenja a to je svojstvo ponovne iskoristivosti ("Library (computing)," 2020).

Svrha korištenja biblioteka u strojnom učenju je olakšati implementiranje algoritama na način da u većini slučajeva ne moramo implementirati logiku rada algoritma, već je dovoljno pozvati određene funkcije i proslijediti joj ispravne parametre i hiperparametre. Ovaj rad se bavi isključivo s otvorenim bibliotekama, a to su one biblioteke koje su besplatne i imaju javno dostupan izvorni kod.

Primjerice, ukoliko se želi implementirati algoritam k-sredina u Python biblioteci scikit-learn, dovoljno je inicijalizirati objekt klase KMeans s željenim brojem klastera i nakon toga pozvati njegovu metodu fit(x), gdje je x polje koje sadrži ulazne podatke. Ako se na kraju želi predvidjeti kojem klasteru neki podatak x pripada, onda je potrebno pozvati metodu predict(x). U nastavku će se navesti neke od najpopularnijih otvorenih biblioteka za strojno učenje.

3.1. Scikit-learn

Scikit-learn (Géron, 2019) je vrlo popularna biblioteka za strojno učenje koja omogućuje niz raznih algoritama iz područja klasifikacije, regresije, klasterifikacije, smanjenja dimenzionalnosti, konfiguracije modela i preprocesiranja. Jedna od njezinih mana je što nju podržava samo programski jezik Python, za razliku od nekih drugih biblioteka koje podržavaju više jezika. Izgrađena je na temelju poznatih Python biblioteka Numpy, SciPy i matplotlib koji služe za matematičke izračune i grafičke prikaze grafova. Koriste ga brojne tvrtke kao što su: J.P.Morgan, Spotify, Change.org, Booking.com, Inria, betaworks.

3.2. Tensorflow

TensorFlow (Hapke et al., 2020) je najpopularnija biblioteka za strojno učenje koju je razvio tim iz Google Brain-a. Ona omogućuje kreiranje grafova podatkovnih tokova odnosno strukture koja opisuje kako se podaci kreću kroz graf. Svaki čvor u grafu predstavlja neku matematičku operaciju, dok svaki rub predstavlja višedimenzionalno polje ili tenzor odakle mu i dolazi ime. Matematičke operacije se izvršavaju u C++-u kako bi se mogle postići visoke performanse, međutim Python se najčešće koristi za pristup API-ju zbog toga što on omogućuje brži i jednostavniji razvoj aplikacija. Za razliku od scikit-learn-a kojeg podržava samo Python, TensorFlow je podržan od strane više jezika kao što su Python, C, C++, Go, Java, JavaScript, Swift. Njegove aplikacije se mogu pokrenuti na mnogo različitih platformi, na primjer na računalu, oblaku (eng. cloud), iOS-u i Androidu-u. Ono što je također vrlo važno za modele s velikim brojem podataka je i mogućnost treniranja ne samo na centralnom procesoru (CPU), već i na grafičkom procesoru (GPU) koji često omogućava brže treniranje. Osim toga, kombinirajući TensorFlow s Google-ovim cloud servisom je moguća upotreba TensorFlow procesnih jedinica (TPU) koje omogućuju još brže treniranje modela. Važno je i reći da TensorFlow koristi Keras-ov API za dodatno olakšavanje rada s neuronskim mrežama.

3.3. Keras

Keras (Atienza, 2020) je biblioteka za rad s neuronskim mrežama koja je napisana u Pythonu. Kao što je prije napisano, ova se biblioteka može koristiti u TensorFlow-u, ali isto tako i u Microsoft Cognitive Toolkit-u, Theano-u, te u programskom jeziku R. Njezina glavna prednost je što je jednostavna za korištenje pa zbog toga ubrzava razvoj modela neuronskih mreža. Ona pokriva sve važne koncepte potrebne pri izradi jednostavnih ili složenijih modela kao što su primjerice slojevi, aktivacijske funkcije, optimizatori i razne normalizacije. Osim standardnih neuronskih mreža, ona omogućuje olakšani razvoj konvolucijskih i rekurentnih neuronskih mreža.

3.4. ML.NET

ML.NET (Capellman, 2020) je Microsoftova biblioteka za strojno učenje napravljena za programske jezike C# i F#. Ukoliko se koristi zajedno s Python modulom NimbusML, tada je također moguća i upotreba modela izrađenih u Pythonu. Kako je ovo još prilično nova biblioteka, ona podržava manji broj algoritama u odnosnu na scikit-learn ili TensorFlow. Prema tome, trenutno se može koristiti za rješavanje problema kao što su: analiza sentimenata, predviđanje cijena, otkrivanje kvarova, prognoziranje prodaje, preporuka proizvoda, segmentacija klijenata i detekcija anomalije, dok se raspoznavanje objekata na slici i klasifikacija slika može koristiti uz pomoć prethodno istreniranih modela. Prednost ML.NET-a je što nudi tzv. *Model Builder* što je jednostavan alat uz pomoć kojeg je jednostavno izraditi prilagođene modele strojnog učenja. On koristi metodu pod nazivom automatizirano strojno učenje koje automatski odabire model s najboljim performansama za zadani problem iz domene strojnog učenja.

3.5. PyTorch

PyTorch (Stevens et al., 2020) je biblioteka za strojno učenje koja se temelji na biblioteci Torch. Razvili su je Facebookovi znanstvenici za umjetnu inteligenciju. Preferirani jezik za upotrebu PyTorcha je Python, no postoji sučelje i za C++ i Javu (za Android upotrebu). PyTorch nudi dvije važne značajke koje omogućavaju lakšu implementaciju algoritama strojnog učenja s visokim performansama. Prva značajka je tenzorsko računanje uz mogućnost upotrebe grafičkog procesora za bolje performanse, a druga značajka je da su duboke neuronske mreže izgrađene na metodi automatske diferencijacije. Automatska diferencijacija služi za snimanje izvedenih operacija kako bi se kasnije prilikom izračuna gradijenata kod širenja unatrag one mogle ponoviti te tako ubrzati vrijeme treniranja neuronske mreže. U ožujku 2018. godine PyTorch se spojio s popularnom bibliotekom Caffe2, te tako proširio svoje mogućnosti. Slično kao i TensorFlow, PyTorch nudi klasu pod imenom Tensor koja služi za obradu višedimenzionalnih polja. Osim nje, PyTorch nudi module za lakšu izgradnju neuronskih mreža, zatim za automatsku diferencijaciju, te za optimizacijske algoritme.

3.6. Apache MXNet

Apache MXNet (Hodnett and Wiley, 2018) je biblioteka za duboko učenje razvijena od tvrtke Apache. Kao i većina ostalih biblioteka, najbolje ju podržava Python, a također postoji podrška i za programske jezike Java, Scala, Julia, Clojure, C++, R i Perl. Ova biblioteka je fleksibilna i podržava simboličko programiranje jednostavnim pozivanjem funkcije *hybridize*. Simboličko izvršavanje omogućuje brži i optimiziraniji rad te mogućnost spremanja i upotrebe modela u nekom drugom jeziku kojeg podržava MXNet. Osim toga, MXNet pruža efikasno distribuirano treniranje modela s više grafičkih procesora sa gotovo linearnim skaliranjem.

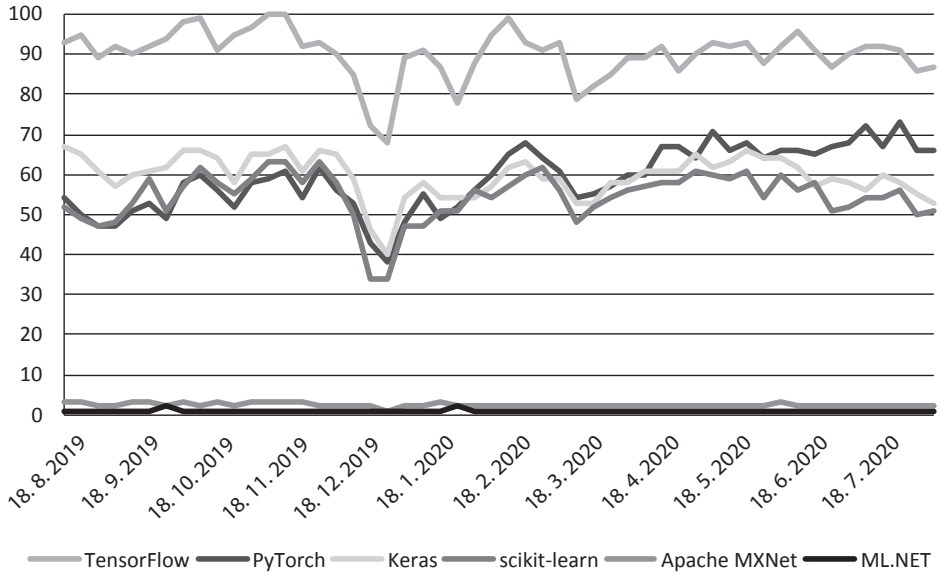
3.7. Usporedba otvorenih biblioteka za strojno učenje

Biblioteke za strojno učenje s kojima smo se ranije upoznali se razlikuju po nekoliko faktora kao što su funkcionalnosti, popularnost, performanse i podržani programski jezici. Biblioteka

scikit-learn ima veliku širinu pokrivenih algoritama i vrlo je jednostavna za korištenje, no najčešće se ne koristi za implementaciju složenijih algoritama kao što su duboke neuronske mreže. Osim toga, za razliku od recimo TensorFlowa i PyTorcha, scikit-learn ne podržava upotrebu GPU-a, što zapravo i nije od velike važnosti zbog toga što se koristi samo za jednostavnije algoritme koji ne zahtijevaju jake računalne performanse. Keras i MXNet se isključivo koriste za rad s neuronskim mrežama, s time da se Keras razlikuje po tome što je on uključen u jezik R i TensorFlow, dok je MXNet biblioteka koja se koristi zasebno. ML.NET je još prilično nova biblioteka pa zbog toga podržava nešto manje algoritama u odnosu na neke druge biblioteke, ali zato omogućuje implementaciju algoritama uz pomoć *Model Buildera*.

Popularnost pojedine biblioteke je moguće odrediti na više načina. Vjerojatno jedan od najvjerodostojnijih i najpouzdanijih načina za provjeriti ovu statistiku je uz pomoć Google trendova koji daju rezultate o kvantitativnoj razini web upita na Google pretraživaču koji su skalirani od 0 do 100. Na slici 2 je vidljivo kako TensorFlow čitavo vrijeme dominira s popularnošću, a slijede ga PyTorch, Keras i scikit-learn. Apache MXNet i ML.NET su znatno manje popularni i teško je za očekivati da će se situacija u bližoj budućnosti promijeniti s obzirom da su ostale navedene biblioteke postale standard u strojnom učenju.

Slika 2. Popularnost biblioteka za strojno učenje



Izvor: https://trends.google.hr/trends [pristupano 15.08.2020.]

Svaka biblioteka za strojno učenje ima za cilj podržati određenu skupinu programskih jezika. Na primjer, scikit-learn podržava samo Python koji je najčešće korišten jezik za strojno učenje kojeg podržava većina biblioteka. Za razliku od scikit-learna-a, TensorFlow i Apache MXNet podržavaju veliku skupinu programskih jezika što omogućuje korištenje strojnog učenja na više različitih platforma kao što su web i pametni uređaji. ML.NET se izdvaja od ostalih biblioteka zbog toga što je njihova ciljna skupina programskih jezika iz Microsoftove

.NET domene (C# i F#), pa je zbog toga teško za očekivati da će u budućnosti podržavati jezike koji nisu u toj domeni. Iz sljedeće tablice vidljivo je koje programske jezike podržava pojedina biblioteka za strojno učenje.

Tablica 1. Podržani programski jezici za pojedinu biblioteku strojnog učenja

Biblioteka za strojno učenje	Podržani programski jezici	Ukupno
scikit-learn	Python	1
TensorFlow	Python, C, C++, Go, Java, JavaScript, Swift	7
Keras	Python, R	2
ML.NET	C#, F#	2
PyTorch	Python, C++, Java	3
Apache MXNet	Python, Java, Scala, Julia, Clojure, C++, R, Perl	8

Izvor: vlastita izrada

4. Aplikacija za kolorizaciju crno-bijelih slika

Za slučaj korištenja otvorenih biblioteka za strojno učenje odabrana je aplikacija za kolorizaciju crno-bijelih slika. Cilj ove aplikacije je omogućiti njezinom korisniku odabir željene crno-bijele slike, te ju na jednostavan pritisak gumba kolorizirati. Nakon što aplikacija uz pomoć strojnog učenja, konkretnije autoenkodera kolorizira odabranu sliku, korisniku se prikazuje predviđena kolorizirana verzija slike koju je moguće spremiti.

Slika 3. Aplikacija za kolorizaciju crno-bijelih slika

Izvor: vlastita izrada

4.1. Skup podataka i korištene tehnologije

Skup podataka koji se je koristio pri treniranju i testiranju modela za kolorizaciju crno-bijelih slika je COCO (eng. Common Objects in Context) koji se sastoji od 123 000 slika raznovrsnih kategorija. Slike su različitih rezolucija (dimenzija), pa ih je potrebno prilikom njihovog učitavanja obraditi kako bi sve slike imale rezoluciju 256x256 koja je kompatibilna s modelom ("COCO - Common Objects in Context," n.d.).

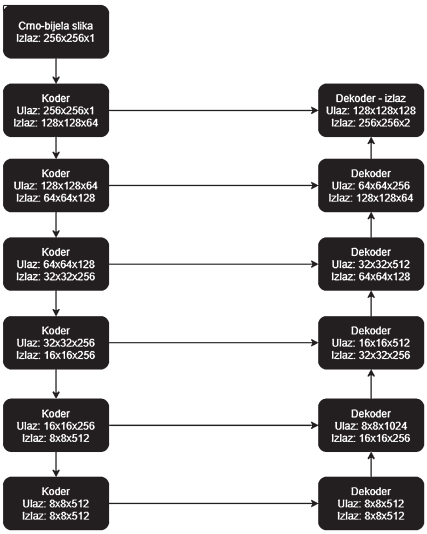
Osim postavljanja istih rezolucija, slike je potrebno pretvoriti u LAB prostor boja. Razlog tome je što prva dimenzija (L) ovog prostora predstavlja svjetlinu čije su vrijednosti izražene od 0 do 100, gdje 0 označava potpuno crnu boju, a 100 bijelu boju. Prema tome, crno-bijela slika se može prikazati uz pomoć prve dimenzije, a preostale dvije dimenzije su izlazni podatak koji se predviđa. Vrijednosti druge i treće dimenzije se najčešće kreću od -128 do 127. Druga dimenzija (A) predstavlja zelenu-crvenu komponentu, gdje negativne vrijednosti označavaju zelenu boju, a pozitivne crvenu boju. Slično, treća dimenzija (B) predstavlja plavo-žutu komponentu, gdje negativne vrijednosti označavaju plavu boju, a pozitivne žutu boju. Prednost korištenja ovog prostora boja je da se ne moraju predviđati tri dimenzije kao što bi to slučaj bio s RGB slikama, već je dovoljno da se predvide dvije dimenzije (A i B). Rezultat spajanja prve dimenzije s preostalim predviđenim dimenzijama je kolorizirana slika u LAB formatu koju je lako moguće pretvoriti u RGB format.

Čitava aplikacija je razvijena u programskome jeziku Python zbog toga što je taj jezik jednostavan i najbolje podržava biblioteke za strojno učenje. Otvorene biblioteke za strojno učenje koje su se koristile pri razvoju aplikacije su Keras i Tensorflow. S obzirom na to da model ima mnogo parametara te skup podataka sadrži mnogo slika, za treniranje modela su se koristili GPU resursi koji su značajno ubrzali proces treniranja. Konkretno, koristio se *Google Colaboratory* koji nudi besplatni pristup GPU i TPU resursima na određeno vrijeme.

4.2. Model autoenkodera

Ulazni podaci modela su crno-bijele slike dimenzija 256x256x1, dok su izlazni podaci slike pretvorene u AB format dimenzija 256x256x2. Kako bismo ubrzali treniranje, na početku skoro svakog sloja se smanjuje širina i visina mape značajki koja je dobivena na prethodnom sloju. Smanjenje je moguće izvesti uz pomoć povećanja broja skokova ili korištenjem sloja sažimanja. Kod problema kolorizacije se bolji rezultati postižu upotrebom skokova, pa se zbog toga na većini konvolucijskih slojeva upotrebljava skok od dva piksela čime se duplo smanjuje širina i visina mape značajki. Svaka funkcija kodera koristi konvolucijske slojeve, dok funkcije dekodera koriste slojeve transponirane konvolucije koje služe za vraćanje slike u izvorni oblik, odnosno za rekonstruiranje slike. Zadnji transponirani konvolucijski sloj mora obavezno imati dva filtera kako bi se na taj način dobio traženi format slike. Model je simetričan što znači da ima jednak broj slojeva za kodiranje i dekodiranje, konkretnije 6 slojeva za kodiranje i 6 slojeva za dekodiranje. Svi slojevi izuzev zadnjeg koriste ReLU funkciju nelinearnosti, a zadnji sloj koristi funkciju tangens hiperbolni kako bi raspon izlaznih vrijednosti bio između -1 i 1. Također, svi slojevi osim zadnjeg spajaju svoj ulaz s pripadajućom izlaznom vrijednošću nasuprotnog sloja. Time model jednsotavnije zadržava komponente originalne slike (Fu and Hsu, 2017). Čitav programski kod je javno dostupan na: https://github.com/pcrncec/Zavrsni_rad.

Slika 4. Model autoenkodera za kolorizaciju crno-bijelih slika



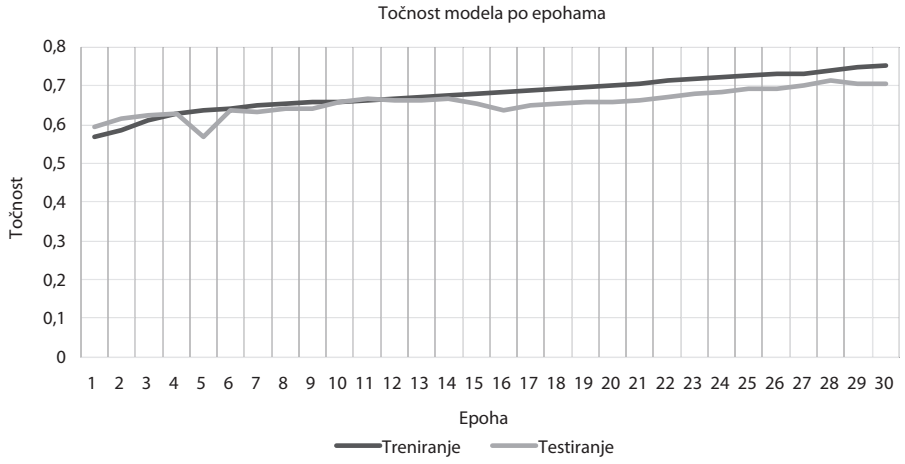
Izvor: vlastita izrada

4.3. Rezultati i evaluacija modela

Kroz proučavanje rezultata kolorizacije moglo se zaključiti da je model uspješniji u predviđanju boja elemenata kao što su nebo, more, zelene površine i ljudi koji se često pojavljuju na slikama te ih je lako za prepoznati. Lošiji je u predviđanju elemenata čija je distribucija manja te onih elemenata koji se na slici pojavljuju na neprepoznatljiv ili neuobičajen način. Rezultati bi vrlo vjerojatno bili bolji da se model trenirao na punom većem skupu podataka, npr. na nekoliko milijuna slika, te da je on bio složeniji što bi mu omogućilo prepoznavanje većeg broja objekata. Međutim, treba uzeti u obzir da bi takvo treniranje trajalo tjednima ili čak mjesecima.

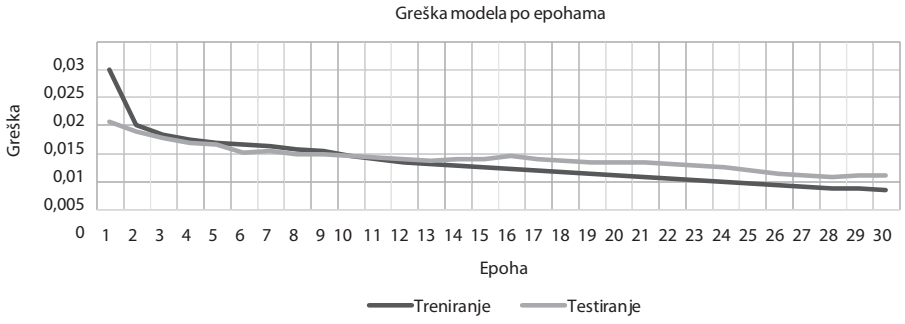
Model je bio treniran na 95% ukupnog broja podataka, a ostalih 5% su bili iskorišteni za testiranje. Točnost modela je kroz epohe konstantno rasla te je na 30. epohi model dostigao točnost od 75.43%. Točnost modela nad podacima za testiranje na kraju treniranja je iznosila 70.37%, što nije loš omjer, odnosno ne postoji problem prekomjernog prilagođavanja (engl. overfitting). Greška modela je konstantno padala te je na kraju treniranja iznosila 0.35%, dok je greška nad podacima za testiranja iznosila 0.62%. Postoje dva razloga zašto je prekinuto treniranje nakon 30 epoha. Prvi razlog je što je samo treniranje ovog modela vremenski zahtjevno, a *Google Colaboratory* ne dopušta dugotrajno treniranje modela (više od 12 sati), pa je sam proces treniranja bio otežan ovim ograničenjem. Drugi razlog koji je presudio u odluci je taj što je model konvergirao, te se je greška modela počela povećavati. U slučaju da se model nastavio trenirati, razlika između preciznosti treniranja i testiranja bi bila sve veća te bi zbog toga nastao problem prekomjernog prilagođavanja.

Slika 5. Točnost modela po epohama



Izvor: vlastita izrada

Slika 6. Greška modela po epohama



Izvor: vlastita izrada

5. Zaključak

Otvorene biblioteke za strojno učenje služe kako bi se olakšala implementacija različitih algoritama strojnog učenja. Od korisnika ne zahtjevaju dubinsko znanje rada algoritma, pa se on može više posvetiti problemima, kao što je primjerice postavljanje optimalnih postavki modela za određeni problem. Biblioteke se razlikuju po parametrima kao što su podrška programskih jezika, performanse, pokrivenost algoritama i popularnost.

Primjena strojnog učenja na problemu kolorizacije crno-bijelih slika može znatno ubrzati i olakšati proces kolorizacije. Međutim, to je vrlo izazovan problem zbog toga što kolorizirane slike sadrže eksponencijalno više kombinacija boja u odnosu na crno-bijele slike, pa je prema tome teško odrediti boju objekta koji se može pojaviti u više različitih boja. Jedan od najboljih pristupa za kolorizaciju crno-bijelih slika upotrebom strojnog učenja su autoenkoderi koji upotrebljavaju konvolucijske neuronske mreže. Uz pomoć njih se mogu rekonstruirati slike na način da se u koderu nauče značajke slika, a u dekoderu se na temelju značajki i predviđenih

boja značajki kolorizira slika. Jedna od mana ovog pristupa je što rezolucija izlazne kolorizirane slike nije dinamična, što znači da će izlazna rezolucija uvijek biti jednaka onoj koja je zadana na specifikaciji modela. Također, još jedna mana korištenja strojnog učenja, konkretnije konvolucijskih neuronskih mreža je što postoji problem kolorizacije onih slika koje sadrže manje poznate objekte. Djelomično rješenje može biti ili upotreba već ranije uspješno istreniranog modela na velikom skupu podataka ili pak povećanje skupa podataka te ponovno treniranje na njemu, no i takve alternative neće riješiti problem u potpunosti. Prema tome, najbolje rješenje problema kolorizacije bi bio hibridni pristup. Najprije bi se uz pomoć aplikacije kolorizirala crno-bijela slika, a nakon toga bi čovjek uz pomoć alata za obradu slika ispravio nepravilnosti i očite greške koje je prouzročila aplikacija. U budućim radovima problem automatske kolorizacije crno-bijelih slika može se probati poboljšati upotrebom različitih arhitektura neuronskih mreža, te duljim treniranjem modela na GPU i TPU resursima u računalnim oblacima.

LITERATURA

1. Alpaydin, E. (2010), Introduction to machine learning, 2nd ed. ed, Adaptive computation and machine learning. MIT Press, Cambridge
2. Atienza, R. (2020), Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, deep RL, unsupervised learning, object detection and segmentation, and more, 2nd Edition, 2nd edition. ed. Packt Publishing
3. Bishop, C.M. (2006), Pattern Recognition and Machine Learning. Springer, New York
4. Buduma, N., Locascio, N. (2017), Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms, 1st edition. ed. O'Reilly Media, Sebastopol, CA.
5. Capellman, J. (2020), Hands-On Machine Learning with ML.NET: Getting started with Microsoft ML.NET to implement popular machine learning algorithms in C#, 1st edition. ed. Packt Publishing
6. COCO - Common Objects in Context [WWW Document], n.d. URL https://cocodataset.org/#home (pristupano 11.10.2020).
7. Fu, Q., Hsu, W.-T. (2017), Colorization Using ConvNet and GAN [WWW Document]. URL /paper/Colorization-Using-ConvNet-and-GAN-Fu-Hsu/327f96c410ab390b2778ffb579d89632b210d337 (pristupano 11.10.2020).
8. Géron, A., (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media
9. Hapke, H., Nelson, C., Safari, O.M.C. (2020), Building Machine Learning Pipelines, O'Reilly Media
10. Hodnett, M., Wiley, J.F. (2018), R Deep Learning Essentials: A step-by-step guide to building deep learning models using TensorFlow, Keras, and MXNet, 2nd Edition. Packt Publishing
11. Library (computing) (2020), https://en.wikipedia.org/wiki/Library_(computing). Wikipedia.
12. Nasteski, V. (2017), An overview of the supervised machine learning methods. HORIZONS.B 4, https://doi.org/10.20544/HORIZONS.B.04.1.17.P05, str. 51–62

13. Oreški, D., Andročec, D. (2018), Hybrid Data Mining Approaches for Intrusion Detection in the Internet of Things, in: 2018 International Conference on Smart Systems and Technologies (SST). Presented at the 2018 International Conference on Smart Systems and Technologies (SST), https://doi.org/10.1109/SST.2018.8564573, str. 221–226
14. Stevens, E., Antiga, L., Viehmann, T. (2020), Deep Learning with PyTorch: Build, train, and tune neural networks using Python tools, 1st edition. ed. Manning Publications
15. Team, E.S. (2020), What is Machine Learning? A definition. Expert Syst. URL https://expertsystem.com/machine-learning-definition/ (accessed 11.10.20).

Summary

APPLICATION OF MACHINE LEARNING TO THE PROBLEM OF COLORIZATION OF BLACK AND WHITE PICTURES

This paper deals with machine learning and its use for the purpose of creating an application for coloring black and white images. Open machine learning libraries were used to help develop the application. The first part of the paper explains machine learning, its most important categories, and its main algorithms. Some of the most popular open machine learning libraries are also explained and compared with each other. The second part explains the implementation of the developed application and the machine learning algorithm that was used in its development. Finally, the results of the application are presented together with the evaluation of the implemented model.

Keywords: *machine learning, libraries, black and white images, colorization, autoencoder.*