TEHNIČKO VELEUČILIŠTE U ZAGREBU

STRUČNI STUDIJ RAČUNARSTVA

Stjepan Salopek

JMBAG: 0246084632

PREPOZNAVANJE TEKSTA RAČUNALNIM VIDOM

ZAVRŠNI RAD br. broj

Zagreb, rujan, 2021.

Sažetak

Seminarski rad metodološki će prikazati izradu aplikacije prepoznavanja teksta računalnim vidom. Pojam računalnog vida obraditi će se kroz tehnologiju umjetne inteligencije, točnije strojnog učenja prolaskom kroz potencijal područja te tržišnu vrijednost. Obraditi će se teorija vezana za kreiranje aplikacije strojnog učenja i navesti primjeri iz suvremenog svijeta u kojima se ono može koristiti.

Nadalje, ukratko će se opisati razne tehnologije koje služe u izradi projekata strojnog učenja, a detaljnije obrazložiti odabrane. Završni dio rada svo navedeno znanje prikazati će u implementaciji programskog kôda, a prikazati će se i konačni rezultati točnosti.

Ključne riječi: strojno učenje, neuronske mreže, prepoznavanje teksta

SADRŽAJ

Sažetak 1
SADRŽAJ 2
Popis tablica 4
Popis slika 5
Popis prikaza programskog kôda 6
1 UVOD 7
2 O UMJETNOJ INTELIGENCIJI 8
2.1 Osnovne definicije 8
2.2 Strojno učenje na tržištu 8
2.3 Potreba za strojnim učenjem 10
2.4 Implementacija strojnog učenja neuronskim mrežama 11
2.5 Vrste neuronskih mreža 13
2.6 Primjeri iz prakse 14
3 STVARANJE APLIKACIJE 16
3.1 Odabir tehnologija 16
3.1.1 Python programski jezik 16
3.1.2 Jupyter Bilježnice 16
3.1.3 Colaboratory 16
3.1.4 TensorFlow 16
3.1.5 PyTorch 17
3.2 Teorija i funkcionalnosti potrebni za izradu 18
3.2.1 Tenzori 18
3.2.2 Uobičajena propagacija neuronske mreže 18
3.2.3 Unazadna propagacija neuronske mreže 19
3.2.4 Stvaranje klase za strojno učenje 20
3.2.5 Implementiranje klase za strojno učenje u Pythonu 21
3.3 Rad sa kreiranom neuronskom mrežom 23
3.4 Rad i analiza rezultata 24
4 PRAKTIČNI ZADATAK 25
4.1 Plan rada 25
4.2 Postavljanje razvojnog okruženja 25
4.3 Preuzimanje podatkovnog skupa 26
4.4 Priprema podatkovnog skupa 26

4.4.1 Funkcija za obradu slike 26
4.4.2 Filtriranje i konačna obrada 28
4.5 Dizajniranje neuronske mreže 29
4.6 Učitavanje podataka 30
4.7 Postavljanje varijabli 30
4.8 Petlja treniranja 32
4.9 Petlja testiranja 33
4.10 Pregled i analiza 34
4.10.1 Matplotlib 34
4.10.2 TensorBoard 34
4.11 Analiza rezultata 36
5 ZAKLJUČAK 37
Popis literature 38

Popis tablica

Tablica 1 Srednje prosječne plaće inženjera strojnog učenja i podataka po državama prema podacima iz 2017. godine 9
Tablica 2 Konačni rezultati treniranja 36

Popis slika

Slika 1 Odnos umjetne inteligencije i strojnog učenja 8
Slika 2 Promjene u plaći inženjera strojnog učenja i podataka po državama uspoređeno sa istima prije tri godine 10
Slika 3 Primjer prepoznavanja brojeva neuronskom mrežom 12
Slika 4 Primjer jednostavne *feedforward* neuronske mreže 13
Slika 5 Primjer prepoznavanja lica strojnim učenjem 14
Slika 6 Primjer strojnog učenja u Google tražilici 15
Slika 7 Usporedba Google trendova za pojmove TensorFlow (crveno) te PyTorch (plavo) ... 17
Slika 8 Primjer uobičajene propagacije mreže 19
Slika 9 Ovisnost varijabli neuronske mreže 20
Slika 10 Model za prepoznavanje teksta 21
Slika 11 Rad u Colaboratoryju 25
Slika 12 Primjer slika u skupu "Handwriting Recognition" 26
Slika 13 Obrada podatkovnog skupa 28
Slika 14 Otvaranje slika bibliotekom Matplotlib 34
Slika 15 Primjer TensorBoard alata 35

Popis prikaza programskog kôda

Prikaz programskog koda 1 – Primjer klase neuronske mreže 21
Prikaz programskog koda 2 – Primjer optimizacijske funkcije 23
Prikaz programskog koda 3 – Pojednostavljeni prolazak kroz mrežu 23
Prikaz programskog koda 4 – Računanje točnosti mreže 24
Prikaz programskog koda 5 Funkcija "prepare_img" 27
Prikaz programskog koda 6 Klasa neuronske mreže 29
Prikaz programskog koda 7 Učitavanje i transformacija slika 30
Prikaz programskog koda 8 Prikaz pkbar statistike 31
Prikaz programskog koda 9 Pripremanje mreže na treniranje 31
Prikaz programskog koda 10 Petlja treniranja mreže 32
Prikaz programskog koda 11 Petlja testiranja istreniranog modela 33
Prikaz programskog koda 12 Primjer TensorBoard rada 35

1 UVOD

U seminarskom radu obrađuje se računalno prepoznavanje teksta. Računalno ili strojno učenje je podijeljeno na više razina apstrakcije i svaka je razina do određene mjere obrađena.

Uvod u područje strojnog učenja kao podvrste umjetne inteligencije započet je definiranjem pojmova korištenih kroz seminarski rad. Nakon navođenja osnovnih definicija, analizira se tržišna vrijednost strojnog učenja i budućeg potencijala u svijetu tako da se pokuša ukratko pojasniti razlog sve veće potražnje i interesa budućih inženjera unatoč velikoj količini.

Nakon toga strojno učenje predstavlja se kao rješenje na probleme iz prakse koje čovjek teoretski može riješiti, no oduzima previše vremena, zbog čega pokušava poslove ranije nedostupne računalima učiniti efikasnijima. Nakon toga obrađuje se pojam neuronskih mreža sa svojim modelima kao način implementacije strojnog učenja. Objašnjen je način rada neuronskih mreža te su navedeni razni primjeri za koje se određeni model koristi. Prije prelaska na praktični dio seminarskog rada, obrađuju se primjeri iz svakodnevnog života u kojima se mogu uočiti aplikacije strojnog učenja.

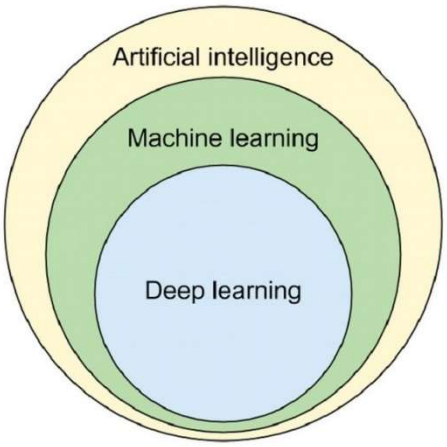
Nakon analize stanja tržišta, definiranja pojmova i uvođenjem u neuronske mreže, rad prelazi na praktičniji dio, odnosno samu implementaciju aplikacije za prepoznavanje teksta programskim jezikom Python koristeći teoriju navedenu kroz rad.

Praktični dio opisuje plan i provedbu projekta, navodi odabrane tehnologije i alate za rad i prikazuje relevantni programski kôd. Nakon toga navodi se uspješnost predviđanja teksta.

2 O UMJETNOJ INTELIGENCIJI

2.1 Osnovne definicije

S obzirom na to da je računarstvo područje koje se iznimno brzo mijenja, sami pojmovi umjetne inteligencije i strojnog učenja često se isprepliću i same definicije ovise o izvoru. Za svrhu seminarskog rada koristiti će se sljedeće općeprihvaćene definicije. Umjetna inteligencija (AI) je grana računarstva koja teži tome da dizajnira sisteme koji obrađuju informacije na sličan način na koji ljudi razmišljaju, a strojno učenje (ML) je jedno od područja istraživanja umjetne inteligencije koje pomoću specijaliziranih algoritamskih postupaka konstruiranih na temelju ljudskih misaonih, odnosno kognitivnih sposobnosti otkriva znanje. [1] [2] Nadalje, računalni vid je interdisciplinarno znanstveno područje koje se bavi načinom na koji računala mogu dobiti višu razinu razumijevanja digitalnih slika ili videozapisa, a neuronske mreže u računarstvu su skupovi logičkih neurona koji uzimaju informacije, obrađuju i zatim prosljeđuju daljnjim neuronima, čime se tvori mreža. [3] [4]



Slika 1 Odnos umjetne inteligencije i strojnog učenja

Može se reći da je umjetna inteligencija, ili točnije strojno učenje zapravo način implementiranja računalnog vida, a implementira se korištenjem neuronskih mreža kroz određeni programski jezik.

2.2 Strojno učenje na tržištu

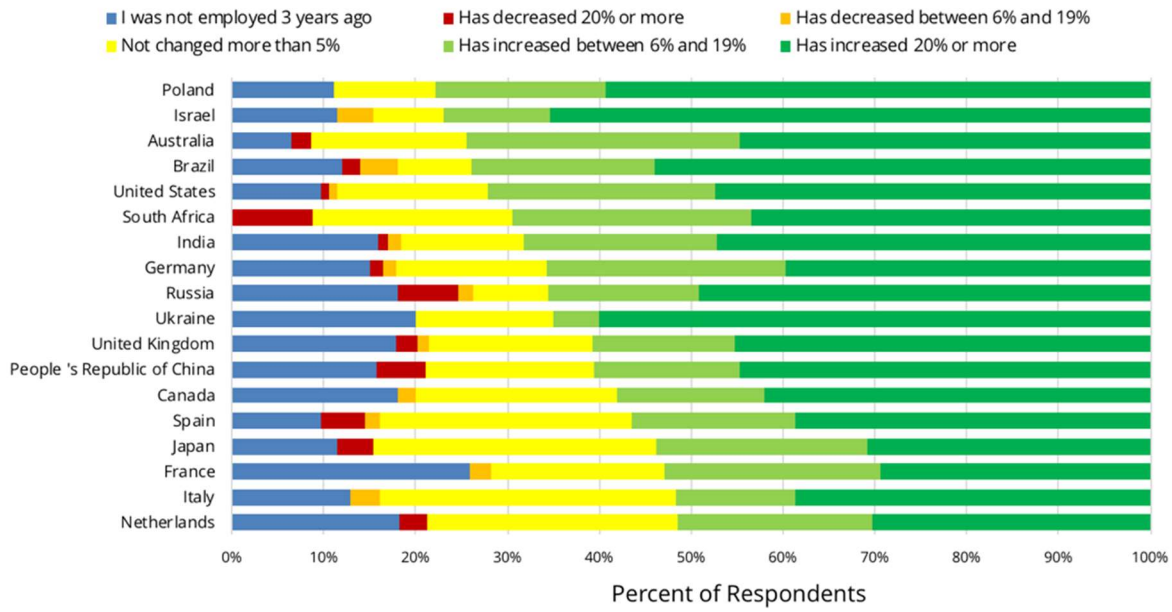
Strojno učenje još uvijek nije zaživjelo na tržištu poput mrežnih aplikacija, no i dalje je snažan alat koji se može koristiti u mnogim područjima. Prema podacima mrežne stranice *indeed*, prosječna plaća inženjera strojnog učenja u SADu je do sto pedeset tisuća dolara godišnje, što je preračunato sedamdeset i osam tisuća kuna mjesečno. [5]

Država	Godišnja plaća izražena u dolarima
Sjedinjene Američke Države	120.000
Australija	110.719
Izrael	87.500
Kanada	81.330
Njemačka	80.120
Nizozemska	75.337
Japan	70.132
Ujedinjeno Kraljevstvo	66.209
Italija	59.791
Francuska	55.008
Južnoafrička Republika	50.051
Španjolska	47.833
Kina	41.310
Brazil	35.349
Poljska	29.003

Tablica 1 Srednje prosječne plaće inženjera strojnog učenja i podataka po državama prema podacima iz 2017. godine [6]

Kao što se može vidjeti u tablici, vodeća tržišta strojnog učenja su razvijenije države poput Sjedinjenih Američkih Država, Kanade, Njemačke, Nizozemske i tako dalje. Razlog tome može biti taj što tržište strojnog učenja još uvijek nije rasprostranjeno poput nekih drugih tehnologija. Osim toga, inženjeri strojnog učenja skuplji su zbog relativne težine područja. Iz tih, ali i mnogih drugih razloga, strojno učenje sveprisutno je u većinom u ogromnim tvrtkama poput Googlea i Facebooka, dok manje tvrtke veću razinu strojnog učenja ne mogu priuštiti ili ista nije isplativa.

Osim trenutne tržišne vrijednosti, strojno učenje ima mnogo dugoročnog potencijala, kako u većim tako i u manjim državama. Grana strojnog učenja svojevrstan je spoj računarstva i matematike, što donekle sužava popis ljudi koji bi se njome mogli baviti. Stoga bi analiza i prikaz budućim inženjerima mogla biti dobar poticaj za odabir domene umjetne inteligencije.



Slika 2 Promjene u plaći inženjera strojnog učenja i podataka po državama uspoređeno sa istima prije tri godine [6]

Osim podataka navedenih u drugoj tablici, zanimljiv podatak je da je najveći porast plaća, prikazan na drugoj slici, češći u srednje ili niže razvijenim državama negoli u razvijenijima. To je dokaz da unatoč dominiranju određenih država na tržištu svejedno postoji ogroman potencijal za inženjere strojnog učenja u budućnosti u većini država, što zbog primjenjivosti znanja inženjera unutar vlastite države, što zbog prirode cijele branše računarstva, odnosno mogućnosti rada na daljinu. To omogućava radnicima iz niže razvijenih država da rade za plaću koja bi bila prosječna, na primjer u Americi, a koja je iznimno dobra u njihovoj vlastitoj državi. Rad na daljinu mogao bi biti ostvaren kroz direktan rad za firme ili kroz sve popularniji način rada, a to je takozvani samostalni rad na daljinu (eng. *freelancing*).

2.3 Potreba za strojnim učenjem

Ljudi često analiziraju podatke da bi iz njih izvukli relevantne informacije. Na primjer, upravitelj tvrtke može kroz tablicu pročitati informacije o poslovanju firme u prethodnom kvartalu i, ovisno o rezultatima, zaključiti posluje li firma dobro ili loše. Vlada određene države može prema tablici nataliteta vidjeti kakvog učinka imaju demografske mjere. No odakle dolaze takve tablice i informacije? U jednostavnim slučajevima mogu doći od osoba, no zbog

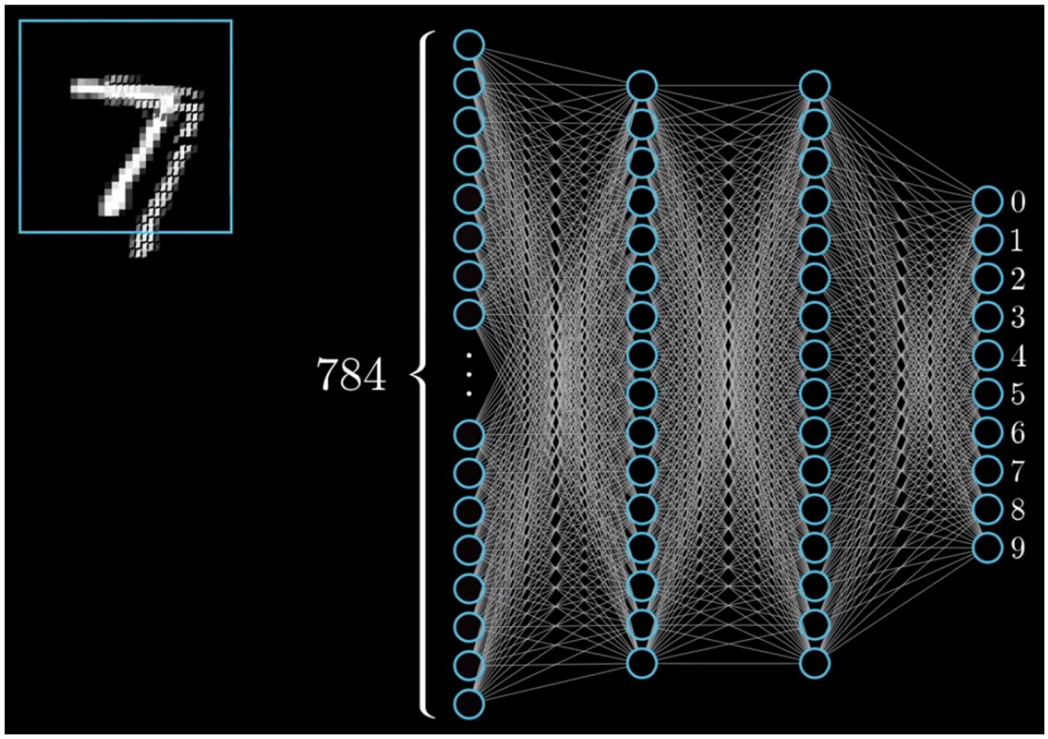
ograničenosti ljudske sposobnosti takvu vrstu informacije obično donosi računalo.

Svijet je prepun informacija. Te informacije mogu biti u formatu medija, na primjer slika, videozapisa i tako dalje, te mogu biti drugog oblika, poput skupa brojeva koji bez analize nemaju posebno značenje. Brojevi sa tablice bez naslova, opisa i oznaka nemaju vrijednost. Milijarde brojeva na jednom mjestu mogu biti stanje bankovnog računa svih osoba područja, brojevi životinja kroz određeni period i tako dalje. Što je više takvih informacija, to je teže čovjeku uspješno dodijeliti značenje na jednostavan način pristupačan publici. Kao u mnogim situacijama, bilo bi idealno kada bi čovjek naporan i repetitivan posao mogao prebaciti na računalo. Upravo na to mjesto dolazi strojno učenje.

2.4 Implementacija strojnog učenja neuronskim mrežama

Strojno učenje može se promatrati kao skup alata i tehnologija koji se mogu koristiti da se pomoću određenih podataka dobiju odgovori na određena pitanja. Na primjer, podaci mogu biti milijuni brojeva, a pitanje može biti koliko efektivne su mjere poboljšanja nataliteta po kvartalima. Osim toga, važno svojstvo strojnog učenja i aplikacija koje ga implementiraju je to da one s vremenom postanu sve točnije. Pri stvaranju aplikacije potrebno je unesti mnogo podataka da računalo pokuša vidjeti inicijalni obrazac podataka i pokuša izmijeniti unutarnje brojke tako da na novim slikama uoči slične obrasce, no osim toga moguće je tokom samog rada aplikacije one vrijednosti koje se unesu na procjenu koristiti za dodatni ispravak unutarnjih brojki tako da se procjena budućih podataka poboljša naknadne procjene.

Da bi se izradila aplikacija strojnog učenja, mora se dizajnirati ili pronaći neuronska mreža koja će se naučiti prepoznavati određene obrasce. Kao što je ranije spomenuto, neuronska mreža je skup povezanih logičkih neurona ili čvorova koja obrađuje ulazne podatke i daje željeni izlaz. Na primjer, ulazni podaci mogu biti karakteristike nekretnine poput broja soba, prozora i kvadrature, a izlazni podatak može biti procjena cijene nekretnine. Postoji mnogo modela mreža, od kojih se dalje mogu dizajnirati različite mreže, a odabir uvelike ovisi o tome što inženjer želi da neuronska mreža nauči predviđati.



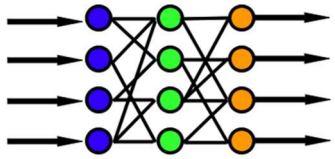
Slika 3 Primjer prepoznavanja brojeva neuronskom mrežom

Neuronske mreže često se objašnjavaju na primjeru prepoznavanja brojeva. Koristeći navedene definicije neuronske mreže, može se uočiti da bi se mreža sa treće slike koristila za prepoznavanje brojeva, da su joj ulazni parametri crno-bijele slike sa različitim nijansama bijele i crne boje, da na ulazu ima broj neurona određen rezolucijom slike, a na izlazu određen i označen znamenkama.

Svi slojevi primaju vrijednost od ulaza ili drugih neurona, vrše neku funkciju i šalju drugim neuronima prema izlazu. Raniji slojevi mreže obično primaju temeljne informacije poput RGB vrijednosti piksela ili skupa piksela, dok kasniji slojevi rade sa apstraktnim informacijama poput linija, rubova, objekata i brojeva. Cilj tokom učenja neuronske mreže jest izmijeniti funkcije koje određeni neuron obavlja na način da na izlazu možemo pogledati vrijednosti neurona označenih znamenkama i vidjeti da je unesena slika broja sedam. Proces učenja uključuje sakupljanje velikog broja slika brojeva, raspodjelom na omjer 70:30 ili 80:20, što bi značilo da će se sedamdeset ili osamdeset posto slika koristiti za unos i izmjenu funkcija neurona, a ostale za provjeru točnosti mreže. Od svih ulaznih informacija unaprijed moramo znati koji broj je unesen tako da se može testirati točnost neuronske mreže.

2.5 Vrste neuronskih mreža

Postoji mnogo vrsta neuronskih mreža koje služe različitim svrhama, od klasifikacije objekata, segmentacije, lociranja pa sve do procjena. [7] U mnogim modelima razlikuju se tri vrste slojeva neurona – jedan ulazni sloj, jedan ili više skrivenih slojeva te jedan izlazni sloj neurona. Neuronske se mreže ponašaju kao crna kutija te se skriveni ili središnji slojevi ne moraju nužno analizirati nakon objave aplikacije. Njihove funkcije obrade mogu se programirano izmijeniti. Osim toga, podaci koji bi se izvukli iz njih ne bi imale veliko značenje za čovjeka jer korisnike zanima samo konačni rezultat na krajnjem sloju, a ne kako je računalo došlo do njega. Osim te zajedničke karakteristike svih neuronskih mreža, one se mogu podijeliti prema načinu obrade informacija. U sljedećem tekstu pri objašnjenjima koristiti će se engleski nazivi modela neuronskih mreža radi očuvanja semantičkog značenja.



Slika 4 Primjer jednostavne *feedforward* neuronske mreže

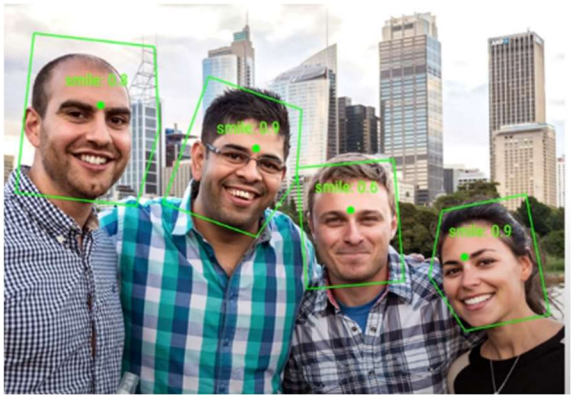
Feedforward neuronska mreža najjednostavniji je model i onaj koji se najčešće uzima kao primjer neuronskih mreža. U takvoj mreži podaci putuju samo unaprijed, prema izlaznom sloju. Feedforward neuronske mreže često koriste mehanizam unazadne propagacije (eng. *back-propagation*) koji označava mijenjanje mreže na temelju izlaznih vrijednosti. Jedna vrsta takve neuronske mreže je konvolucijska neuronska mreža, koja je također duboka neuronska mreža. Ona je vrlo popularna i precizna u radu sa digitalnom grafikom pri detekciji, klasifikaciji i segmentaciji objekata.

Radial basis function (RBF) neuronske mreže uračunavaju udaljenost neurona od određene točke neuronske mreže, a može se koristiti u generiranju aproksimacije matematičke funkcije, predviđanju vremenskih funkcija, klasifikaciji te kontroli dinamičkih sistema. RBF neuronske mreže imaju tri sloja. Ulazni sloj ima po jedan neuron za svaki ulaz, srednji skriveni sloj ima unaprijed neodređen broj, a izlazni sloj neurona jedan je rezultat, a u slučaju klasifikacije je rezultata, odnosno neurona onoliko koliko je kategorija, a ti rezultati predstavljaju vjerojatnosti da ulaz spada pod određenu kategoriju.

Recurrent neuronske mreže slične su prvoj navedenoj vrsti, no razlika je što u njima informacije mogu putovati u oba smjera, odnosno isti neuron i njegova memorija može se koristiti za obradu veće količine podataka. Zbog tog svojstva obično se primjenjuju u primjeni prepoznavanja rukopisa ili prepoznavanja govora.

Modularne neuronske mreže najsličnije su čovjekovoj neuronskoj mreži jer, poput čovjekove, implementirane su segmentacijom i modularizacijom težih zadataka u manje. [8] Na primjer, ljudski mozak vid paralelno raščlanjuje i obrađuje na boju i na kontrast u slojevima bočne genokulatne jezgre, a nakon toga se rezultati sažimaju u drugim dijelovima mozga. [9] Modularne neuronske mreže nisu popularne kao na primjer konvolucijske, no razvojem sklopovlja povećavati će se njihov potencijal u prostoru razumijevanja bioloških neuronskih mreža. Na primjer, moglo bi se razviti bolje razumijevanje toga kako primati raščlanjuju problem hvatanja objekata i zašto to rade vrlo efektivno u odnosu na druge životinje.

2.6 Primjeri iz prakse

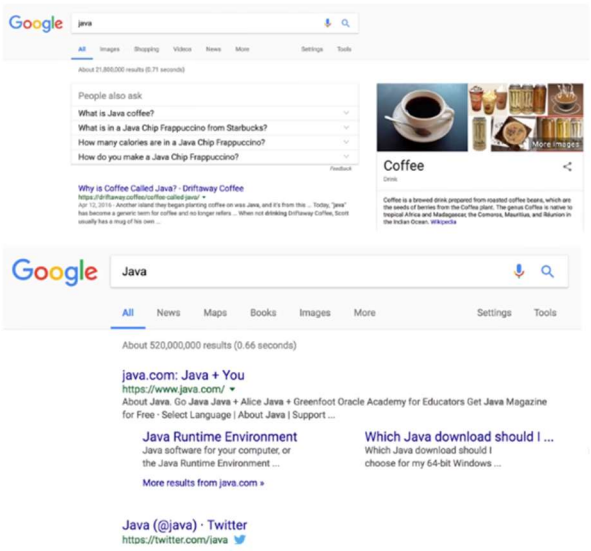


Slika 5 Primjer prepoznavanja lica strojnim učenjem

Kao što je ranije spomenuto, strojno učenje češće je u velikim tvrtkama zbog cijene strojnih inženjera i težine implementacije, a usmjereno je na poboljšanju korisničkog iskustva i kvalitete usluge. Facebook već dugo vremena koristi tehnologiju strojnog učenja za detekciju lica, kojom se može omogućiti automatsko predlaganje osoba na fotografiji za označavanje. Osim toga, strojno učenje se koristi za predlaganje prijatelja na temelju informacija kao što su lokacija osobe, ostali prijatelji, interesi i sviđanja drugih stranica, filmova i tako dalje. [10]

U Googleu strojno učenje koristi se za mnogo stvari, a najpopularnija implementacija je mrežna tražilica. Ona strojno učenje koristi za poboljšano rangiranje podataka tako da korisnik ranije pronađe željene podatke. Alat za to,

osim samog upita upisanog u tražilicu, koristi i informacije koje može dobiti o korisniku iz njegove IP adrese, korisničkog računa, drugih pretraga korisnika njegovog područja u to doba i još mnogo toga. Na primjer, na šestoj slici može se uočiti da isti upit može dati potpuno različite rezultate. U prvoj polovici slike korisnik je pretragom izraza „java“ dobio vrstu kave, dok je na drugoj slici različit korisnik dobio za taj upit dobio programski jezik Java. Iz toga se može pretpostaviti da potonji često šalje upite vezane za računarstvo, dok prvi češće šalje upite vezano za kofeinske napitke.



Slika 6 Primjer strojnog učenja u Google tražilici

Na navedenom primjeru može se vidjeti koliko je strojno učenje korisno jer, u slučaju da je korisnik usluge znanstvenik koji traži relevantnu literaturu, mnogo vremena može se uštediti tako što će tražilica pokušati predvidjeti koje znanstvene radove ili članke on traži.

3 STVARANJE APLIKACIJE

3.1 Odabir tehnologija

U sljedećim odlomcima obraditi će se razne tehnologije i alati koji se mogu koristiti pri izradi aplikacije prepoznavanja teksta, te navesti i obrazložiti konačni odabir istih.

3.1.1 Python programski jezik

Za stvaranje aplikacije strojnog učenja koristi se Python programski jezik koji je jedan od najpopularnijih kada je u pitanju umjetna inteligencija. Unutar jezika postoje mnogi paketi namijenjeni za strojno učenje. Ukratko će se navesti jedni od najpopularnijih paketa ili biblioteka - TensorFlow i PyTorch. Potonji je odabran za izradu seminarskog rada zbog jednostavnosti uporabe i intuitivne izvedbe.

3.1.2 Jupyter Bilježnice

Jupyter Bilježnice (eng. *Jupyter Notebooks*) su razvojna okruženja smještena na javnom ili lokalnom serveru koja omogućuju pisanje Python kôda, naredbi komandne linije (eng. *command-line*) i još mnogo toga. Imaju sposobnost prikaza formatiranog teksta poput HTMLa, a često se koriste kao demonstracije funkcionalnosti ili pri obradi velikih podataka, primjerice u strojnom učenju.

3.1.3 Colaboratory

Colaboratory ili Google Colab proizvod je Googlea koji omogućuje udaljeno spajanje na Jupyter Bilježnice. Može se otvoriti u mrežnom pregledniku, a najčešće se koristi za demonstracije Python kôda te u svrhe strojnog učenja. Njegova najveća prednost je što, uz korištenje procesorske snage na Googleovim računalima, omogućuje i besplatan, ali vremenski ograničen pristup grafičkim karticama koje višestruko ubrzavaju treniranje modela neuronske mreže. Iz tog razloga odabran je za razvojno okruženje u kojemu je napisan praktični dio rada.

3.1.4 TensorFlow

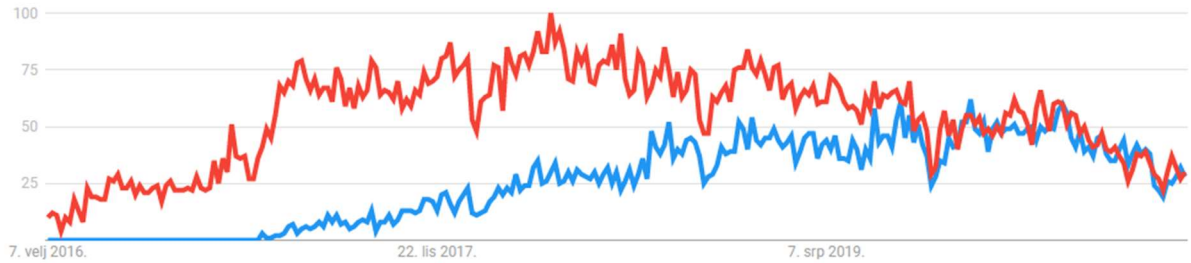
TensorFlow je paket otvorenog kôda namijenjen za numeričke izračune visokih performansi korištenjem grafova protoka podataka (eng. *Data flow graph*). U pozadini TensorFlow je radni okvir koji izvodi računske operacije pomoću tenzora. Prednosti uključuju mnoge algoritme unutar paketa, ogromnu zajednicu,

podršku rada kroz više grafičkih kartica i procesora. Nedostaci su vrlo niža brzina izvođenja u odnosu na druge biblioteke te relativno visoka težina učenja.

3.1.5 PyTorch

PyTorch je popularna biblioteka strojnog učenja za programski jezik Python, baziran na biblioteci Torch implementiranoj u programskim jezicima C i Lua. Izradio ga je Facebook, a koriste ga mnoge velike korporacije poput Twittera. Prednosti uključuju mnoge sposobnosti dubokog učenja i algoritama, korištenje grafičke kartice za ubrzanje računanja te mnogo istreniranih modela. Nedostaci su vezani za relativno mladu dob, a to su manjak resursa na internetu u odnosu na druge biblioteke.

Sedma slika prikazuje trendove pretrage. Može se uočiti da je TensorFlow dugo vremena bio popularniji na tržištu, no da je u zadnje vrijeme otprilike jednako



Slika 7 Usporedba Google trendova za pojmove TensorFlow (crveno) te PyTorch (plavo)

popularan na Google tražilici kao što je to PyTorch.

3.2 Teorija i funkcionalnosti potrebni za izradu

Ranije u radu ukratko je navedena relevantna teorija za seminarski rad. U sljedećim odlomcima detaljnije će se obraditi dijelovi koji su vezani za izradu aplikacije strojnog učenja. Obraditi će se podatkovna struktura tenzora, rad neuronske mreže kroz uobičajenu i unazadnu propagaciju te kreiranje klase koja predstavlja neuronsku mrežu. Rad koristi *feedforward* tip mreže.

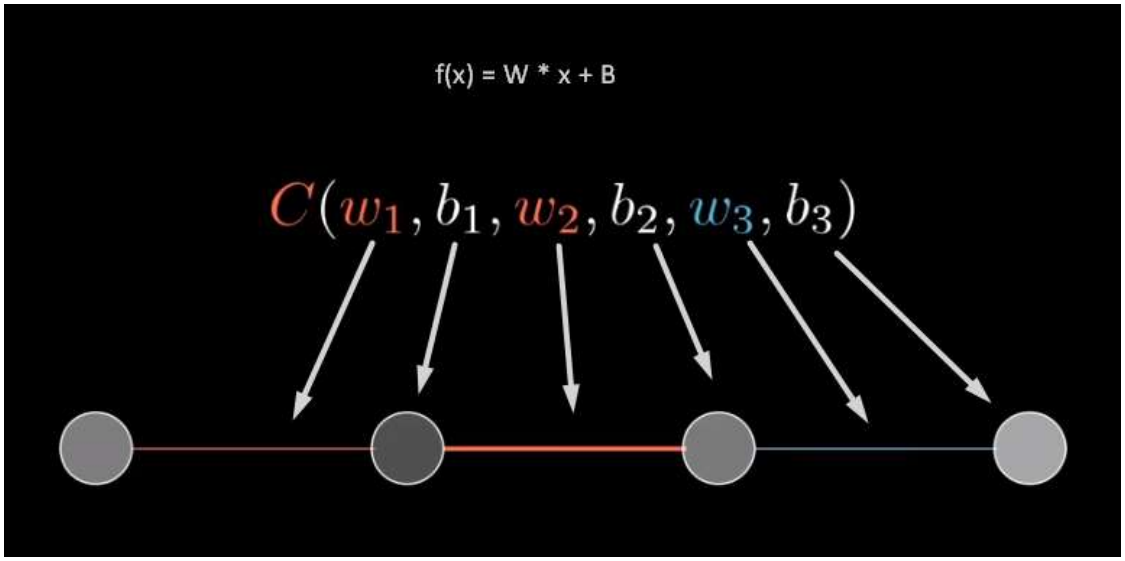
3.2.1 Tenzori

Tenzor je izraz posuđen iz matematike, a u strojnom učenju označava strukturu sličnu matricama više dimenzija. Tenzori u PyTorchu se sastoje od jednog tipa podatka, najčešće od realnog broja širine 16, 32 ili 64 bitova, a može i biti kompleksan broj širine do 128 bitova, *boolean* vrijednost ili cijeli broj širine do 64 bita.

3.2.2 Uobičajena propagacija neuronske mreže

Neuronske mreže ranije su opisane kao mreže u kojima se informacije kreću prema izlazu koji se može promatrati kao sloj jednog ili više neurona na desnom rubu. Neuroni su povezani standardnim linearnim funkcijama oblika $f(x) = a * x + b$.

Parametar „a“ linearne funkcije naziva se težina (eng. *weight*), parametar „b“ označava sklonost (eng. *bias*). Mreža bez ikakvih promjena ima konstantne težine i sklonosti, a mijenjaju se ulazne vrijednosti, odnosno parametar „x“, koji može označavati R, G ili B vrijednost piksela.



Slika 8 Primjer uobičajene propagacije mreže

Na osmoj slici može se vidjeti prikaz jednostavne neuronske mreže sa jednom vezom između dva neurona. U toj mreži, težine su označene linijama između neurona, a sklonosti su konstante na čvorovima. Funkcija $f(x)$ označava prijelaz iz jednog čvora u drugi. Prema toj analogiji, prelazak iz ulaznog čvora, preko dva skrivena, pa sve do izlaznog, jest uzimanje ulaznog parametra „x“ i slijedni prolazak kroz tri funkcije $f(x)$, od kojih svaka ima svoju težinu ($w1$, $w2$, $w3$) i svoju sklonost ($b1$, $b2$, $b3$).

3.2.3 Unazadna propagacija neuronske mreže

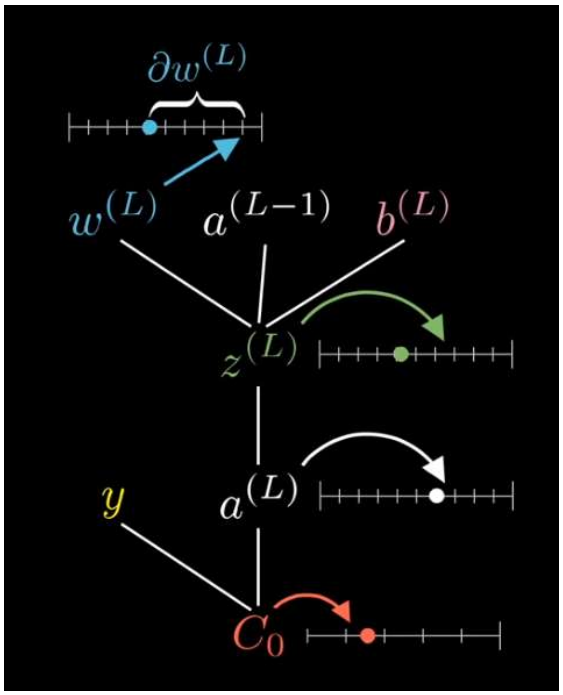
Uobičajena propagacija koristi se za predviđanje kada znamo koje težine i sklonosti treba postaviti na čvorove mreže koju smo dizajnirali. Na primjer, za mrežu sa dva čvora u kojoj je mreža istrenirana da izlaz bude vrijednost 2.5, iz formule $f(x) = W * x + B = 2.5$ možemo zaključiti da je W postavljen na 0, a B na 2.5, što će za svaki „x“ rezultirati u željenom izlazu $f(x) = 0 * x + 2.5 = 2.5$.

Međutim, kada su u pitanju veće mreže, i kada želimo predvidjeti kompleksnije stvari, na primjer nalazi li se na slici mačka ili pas, tada se koristi metoda unazadne propagacije. Slično kao što se u prethodnom odlomku mreža istrenirala za izlaz 2.5 postavljajući težinu i sklonost, tako se unazadna propagacija koristi da se pokuša predvidjeti vrijednost težina i sklonosti u mnogo većim mrežama, i time dobio željeni izlaz.

Prema devetoj slici prikazati će se unazadna propagacija između dva neurona.

Na slici je ulaz u prvi neuron, ili „x“ označen sa $a(L-1)$. Težina prema drugom neuronu je označena sa $w(L)$, a sklonost sa $b(L)$. Slijedi da je $f(x)$ ili ulaz drugog neurona sljedeće: $f(a(L-1)) = w(L) * a(L-1) + b(L)$. Dakle, izlazna vrijednost ovisi o ulaznoj. Da bismo dobili ovisnost izlazne vrijednosti o ulaznoj, moramo derivirati funkciju $f(x)$, čime se dobije gradijent tog čvora. Tada, ukoliko je gradijent malen, male promjene varijable „x“ rezultirati će velikim izlaznim promjenama i obrnuto. Pomoću gradijenta se mijenjanju konstante čvora $w(L)$ i $b(L)$ tako da se dobije izlaz bliži željenoj vrijednosti.

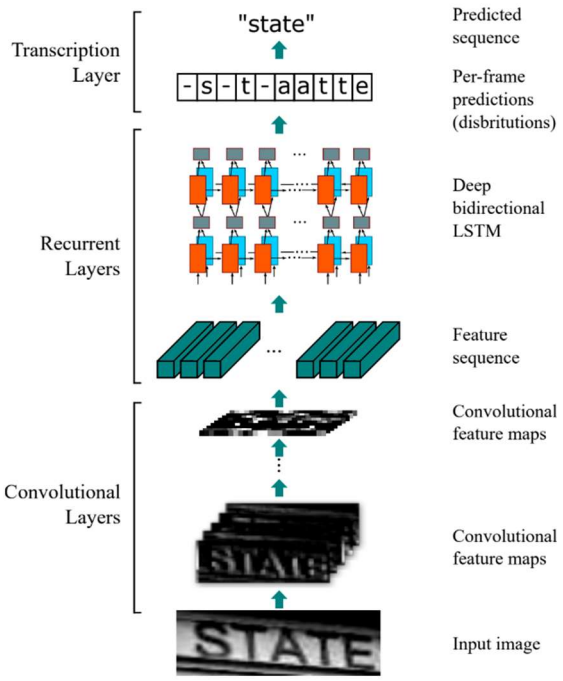
Prolaskom unazadne propagacije kroz cijelu mrežu paket PyTorch automatski sprema derivacije ili gradijente čvorova u njih same, te se na temelju toga može dobiti preciznija izlazna vrijednost.



Slika 9 Ovisnost varijabli neuronske mreže

Unazadna propagacija koristi se u kontekstu treniranja modela, a ne predviđanja teksta. Primjerice, to je slučaj kada se zna koji „x“ ulazi u funkciju $f(x)$ i kada se mogu mijenjati određene karakteristike te funkcije, na primjer težina i sklonost. Nakon što je model istreniran, parametar „x“ smatra se nepoznanicom jer kada je aplikacija objavljena, korisnici aplikacije mogu poslati bilo kakve vrijednosti kao ulaz.

3.2.4 Stvaranje klase za strojno učenje



Slika 10 Model za prepoznavanje teksta

Deseta slika prikazuje popularnu neuronsku mrežu iz znanstvenog rada vezanog na temu prepoznavanja teksta. [11] Model kroz nekoliko koraka uzima sliku, primjenjuje na njoj mnoge konvolucije koje će naglasiti relevantne dijelove, a zatim se ti dijelovi dalje rastavljaju i u konačnici dobijemo predviđanje teksta.

Da bi se navedena neuronska mreža implementirala u PyTorchu, potrebno je uključiti sljedeće programske pakete: torch, torchvision, torchvision.transforms, torch.nn i torch.nn.functional.

3.2.5 Implementiranje klase za strojno učenje u Pythonu

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.konvolucija = nn.Conv2d(...)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)

    def forward(self, x):
        # Obrada kroz mrežu
        x = self.pool(F.relu(self.konvolucija(x)))
        x = torch.nn.functional.relu(self.fc1(x))
        return x
```

Prikaz programskog koda 1 – Primjer klase neuronske mreže

Prikaz pojednostavljene klase na prvom programskom isječku pokazuje radnje potrebne za kreiranje klase. Pri kreiranju klase mora se naslijediti nn.Module koji sadrži mnoge dijelove ranije spomenute u radu koji se automatski odrađuju. Na

primjer, klasa je zadužena za rad unazadne propagacije i podešavanje težina i sklonosti čvorova neuronske mreže.

U konstruktoru klase definiraju se radnje i konvolucije prikazane u modelu desete slike i mogu se postaviti određeni parametri koji će se koristiti u neuronskoj mreži.

Nadalje, mora se dizajnirati „forward“ funkcija koja će predstavljati jedan prolazak jedne ulazne vrijednosti kroz mrežu. Ta funkcija poziva se u pozadini kada se objektu klase pošalje objekt, primjerice slika koja sadrži tekst.

3.3 Rad sa kreiranom neuronskom mrežom

```
import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

Prikaz programskog koda 2 – Primjer optimizacijske funkcije

Da bi se dostigao učinak čitanja teksta iz grafike, potrebno je stvoriti objekt klase neuronske mreže i na njoj primijeniti sljedeće radnje: definirati funkciju koja će mijenjati težine i sklonosti. Implementacija je prikazana na dvanaestoj slici.

Funkcija korištena za optimizaciju težina i sklonosti naziva se Stohastični Spust Gradijenata (eng. *Stochastic Gradient Descent*), ili SGD funkcija. Navedena funkcija težinu mijenja pomoću njene trenutne vrijednosti, gradijenta dobivenog u unazadnoj propagaciji i konstante poslane koja određuje brzinu promjene:

$$weight = weight - learningRate * gradient$$

```
for ponavljanje in range(2):
    for index, podaci in enumerate(spremnik_podataka):

        # Nuliranje prethodnih gradijenata
        optimizer.zero_grad()

        # Prolazak kroz mrežu i optimizacija
        izlaz = mreza(podaci)
        optimizer.step()
```

Prikaz programskog koda 3 – Pojednostavljeni prolazak kroz mrežu

Nakon definiranja funkcije koja vrši optimizaciju, slijedi treniranje. Iz trećeg programskog isječka može se uočiti da treniranje uključuje prolazak po podacima, u ovom slučaju slika sa tekstom i slanjem tih podataka u objekt ranije kreirane koja predstavlja neuronsku mrežu. Proces se odvija dva puta u petlji radi povećanja preciznosti neuronske mreže.

3.4 Rad i analiza rezultata

```
tocni = 0
sveukupno = 0

for podaci in svi_podaci:
    izlaz = mreza(podaci)
    # Usporedba...
    #tocni += 1
    sveukupno += 1

    print("Preciznost: %2d %%" %(tocni / sveukupno * 100))
```

Prikaz programskog koda 4 – Računanje točnosti mreže

Pri usporedbi trinaeste i četrnaeste slike može se primijetiti sličnost. Kada je u pitanju predviđanje, koristeći ulazne slike koje sadrže tekst prolazi se kroz petlju šaljući slike kroz mrežu. Da bi se rezultati mogli analizirati, koriste se slike čiji se tekst unaprijed zna. Nadalje, u dijelu označenim komentarom obrađuje se provjera i povećava brojač točnih rezultata za točne procjene. Na kraju izvedbe pomoću sveukupnog broja slika za predviđanje i broja točnih predviđanja dobiva se točnost neuronske mreže u predviđanju. Suvremene neuronske mreže sa većim brojem ulaznih podataka za testiranje i jakim računalima mogu, ovisno o vrsti slike koja sadrži tekst, dostići vrijednosti između 95 i 100% točnosti.

4 PRAKTIČNI ZADATAK

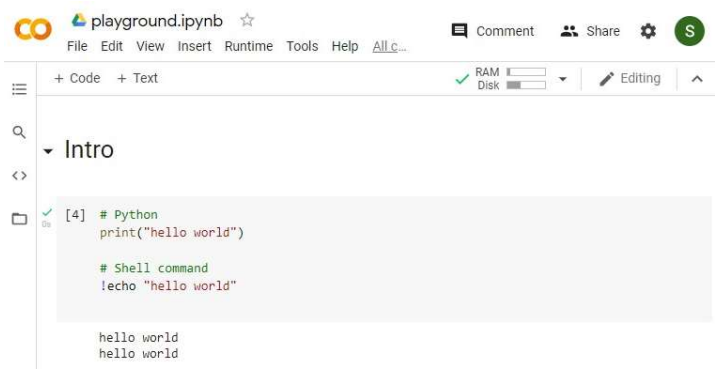
4.1 Plan rada

Nakon opisa i analize strojnog učenja na razini tržišta, teorije i samih tehnologija implementacije, slijedi pisanje plana rada. Pristup problemu je prije svega pragmatičan – podatkovni skup preuzet sa interneta obrađen je, a kroz mnoga testiranja i mijenjanja vrijednosti varijabli pokušala se dostići što veća preciznost procjene.

Početak izrade uključuje postavljanje razvojnog okruženja, nakon čega se preuzima inicijalni podatkovni skup. Nadalje, skup se mora obraditi i prilagoditi kontekstu u kojemu će biti iskorišten. Zatim slijedi dizajniranje neuronske mreže koja će se koristiti za treniranje modela. Istrenirati će se više modela i za svaki testirati preciznost čitanja teksta.

4.2 Postavljanje razvojnog okruženja

Kompletan praktični dio rada odrađen je na Googleovoj platformi. Google Drive korišten je za pohranu podataka, a Colaboratory ili Colab za obradu.



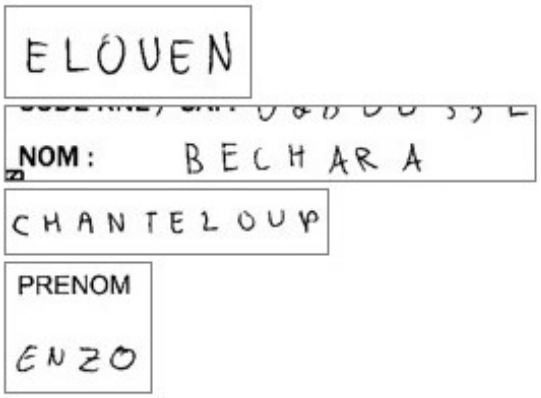
Slika 11 Rad u Colaboratoryju

Colab pojednostavljuje rad jer ima funkcionalnosti poput uvoza Google Drivea direktno u Linux virtualnu mašinu na kojoj je pokrenuta Jupyter bilježnica. Također, mogu se direktno u bilježnici pokretati naredbe ljuske što se pokazalo korisnim u analizi podataka.

Iako Colab omogućuje pristup grafičkim karticama, vremenski su ograničene. Iz tog razloga obrada podatkovnog skupa odrađena je procesorskom snagom, a treniranje neuronske mreže preko grafičke kartice.

4.3 Preuzimanje podatkovnog skupa

Podatkovni skup preuzet je sa Kaggle stranice, a naziva se „Handwriting Recognition“. [12] Sastoji se od tri direktorija u kojima se nalaze slike s imenima za treniranje, testiranje i za validaciju. Za svaki od njih postoji i tablica koja govori koje ime se nalazi na određenoj slici. Sveukupno se u skupu nalazi 400 tisuća slika te 140 tisuća različitih imena. Prilikom analize podatkovnog seta utvrđeni su elementi neprikladni za treniranje. Na dvanaestoj slici može se uočiti da ime može sadržavati dodatni tekst, te da isti može biti pisan različitim stilovima.



Slika 12 Primjer slika u skupu "Handwriting Recognition"

4.4 Priprema podatkovnog skupa

4.4.1 Funkcija za obradu slike

Zbog nepravilnosti spomenutih u prethodnom odlomku i zbog toga što nema mnogo slika za svako ime, skup je obrađen, a model je treniran po svakom slovu, a ne imenu. Prvi korak u obradi je bilo definiranje funkcije „prepare_img“ za dohvat informacija. U funkciji se pomoću „pytesseract“ biblioteke i „tess.image_to_boxes“ metode dobiju koordinate svih slova sa slike.

```
def prepare_img(path, name):
  padding = 3
  pil_img = Image.open(path)
  pil_img = ImageOps.expand(pil_img, border=padding, fill=(255, 255, 255))
  letters = []
  boxes = tess.image_to_boxes(pil_img).upper().split("\n")
  boxes = [item for item in boxes if len(item) > 0]

  for lett in boxes:
    lett = lett.split(" ")
    coords = [ int(item) for item in lett[1:-1]]
    temp = pil_img.height - coords[1]
    coords[1] = pil_img.height - coords[3]
    coords[3] = temp
    x1 = coords[0]
    y1 = coords[1]
    x2 = coords[2]
    y2 = coords[3]
    letters.append({
        'char': lett[0],
        'image': pil_img.crop((
            x1 - padding,
            y1 - padding,
            x2 + padding,
            y2 + padding
        )),
        'coords': (x1, y1, x2, y2)
    })

  return {
      'path': path,
      'name': name,
      'image': pil_img,
      'letters': letters
  }
```

Prikaz programskog koda 5 Funkcija "prepare_img"

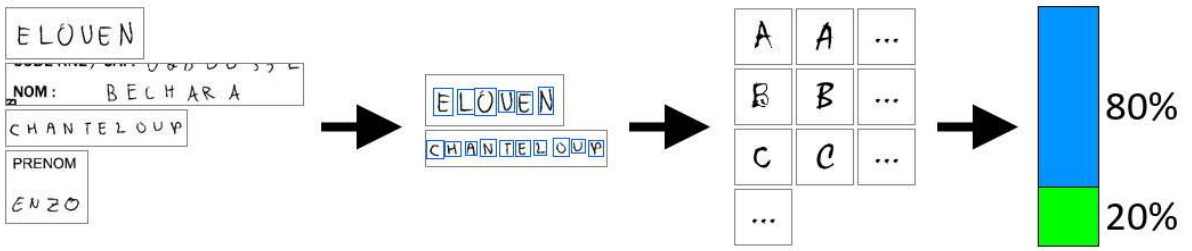
4.4.2 Filtriranje i konačna obrada

Funkcija iz petog isječka programskog kôda vraća osnovne informacije o slici, na primjer koliko se u njoj nalazi slova te gdje se ta slova nalaze.

Te informacije upotrijebljene su za obradu izvornog skupa podataka. Sve slike koje nisu sadržavale isključivo ime izbrisane su. Na primjeru iz trinaeste slike može se uočiti da je slika sa imenom „ENZO“ nestala u sljedećem koraku iz razloga što sadrži i tekst „PRENOM“.

Nadalje, pomoću spomenute funkcije slika je podijeljena na slova. Svako slovo spremljeno je kao zasebna datoteka, a vrijednost slova pročitana je iz imena koje se nalazi u tablici. Na taj način podatkovni skup sa slikama imena pretvoren je u podatkovni skup slova.

U konačnici, slova su podijeljena u direktorij za treniranje mreže te za provjeru mreže. Od svakog slova dvadeset posto nasumično odabranih slika premješteno je za provjeru, a ostatak za treniranje.



Slika 13 Obrada podatkovnog skupa

4.5 Dizajniranje neuronske mreže

Korištena neuronska mreža sastoji se tri linearna sloja između kojih se nalaze dvije aktivacijske funkcije naziva ReLU (eng. *Rectified Linear Unit*) opisane formulom $f(x) = \max(0, x)$. ReLU je odabran radi brzine izvedbe i zbog male šanse za nastanak problema nestajućeg gradijenta. Mreža se sastoji od 784 ulaznih vrijednosti koje označavaju sliku dimenzija 28x28 te 26 izlaznih za svako slovo engleske abecede. Za implementaciju u Pythonu korištena je „torch.nn“ biblioteka.

```
class RecognitionNet(nn.Module):

  def __init__(self, input_size, output_size):
    super(RecognitionNet, self).__init__()
    self.inp = input_size
    self.out = output_size
    self.layer_2 = 128
    self.layer_3 = 64
    self.fc1 = nn.Linear(self.inp, self.layer_2)
    self.fc2 = nn.Linear(self.layer_2, self.layer_3)
    self.fc3 = nn.Linear(self.layer_3, self.out)

  def forward(self, x):
    x = self.fc1(x)
    x = nn.functional.relu(x)
    x = self.fc2(x)
    x = nn.functional.relu(x)
    x = self.fc3(x)
    x = nn.functional.log_softmax(x, dim=1)
    return x
```

Prikaz programskog koda 6 Klasa neuronske mreže

Klasa prikazana u šestom programskom kôdu u konstruktor prima ulazne i izlazne vrijednosti mreže, odnosno ranije navedene vrijednosti 784 te 26. Osim konstruktora, nasljeđivanjem klase „nn.Module“ potrebno je implementirati metodu „forward“ koja je pozvana kada su mreži poslane slike.

Osim toga, neuronska mreža a i neke druge klase mogu biti prebačene na grafičku karticu ukoliko je ista dostupna. To se čini pozivanjem naredbe „.to(device=torch.device('cuda'))“.

4.6 Učitavanje podataka

Pri radu sa podacima PyTorch pruža mnoge mogućnosti preko biblioteka. Za učitavanje kompletnog podatkovnog skupa korištena je funkcija „torchvision.datasets.ImageFolder“. Ona svaku sliku povezuje sa njenim direktorijem. Primjerice, na slike unutar direktorija „A“ postaviti će tu oznaku.

```
data_transforms = transforms.Compose([
    transforms.Resize((28, 28)),
    transforms.Lambda(lambda img: img.convert('L').filter(ImageFilter.S
HARPEN) if sharpen else img.convert('L')),
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
trainset = datasets.ImageFolder('datasets/letters/train', transform=dat
a_transforms)
testset = datasets.ImageFolder('datasets/letters/test', transform=data_
transforms)
```

Prikaz programskog koda 7 Učitavanje i transformacija slika

U programskom isječku mogu se primijetiti i transformacije podataka iz biblioteke „torchvision.transforms“ koje su zadužene za dodatno pripremanje slika za rad. Funkcija „Resize“ svaku sliku prilagodi ranije spomenutoj vrijednosti na ulazu mreže. „Lambda“ je prilagođena funkcija koja dodatno sliku pretvara iz RGB formata u format sa jednim bajtom po pikselu.“ToTensor“ također prilagođava sliku neuronskoj mreži, a „Normalize“ je čini uglađenijom, tako da neuronska mreža ne dobije neočekivane vrijednosti.

4.7 Postavljanje varijabli

Prilikom rada aplikacije koriste se mnoge promjenjive varijable koje uvelike utječu na rad. Objekti „DataLoader“ klase zaslužni su za miješanje slika te za učitavanje više njih istovremeno. Varijabla koja određuje broj istovremeno obrađenih slika naziva se „batch_size“. Osim tih objekata, potrebno je stvoriti i objekt klase neuronske mreže, objekt funkcije gubitka te optimizator. Osim toga spremaju se epohe i vrijeme u svrhu boljeg treniranja i preglednije analize rezultata te stopa učenja koja određuje brzinu mijenjanja težina i sklonosti pri unazadnoj propagaciji. Za dodatnu analizu koristi se „pkbar“ biblioteka koja prikazuje korisne vrijednosti tijekom treniranja:

```
Epoch: 51/100
143/169 [==========>...] - 57s 336ms/step - loss: 0.1331 - acc: 93.8389
```

Prikaz programskog koda 8 Prikaz pkbar statistike

```
train_loader = DataLoader(trainset, batch_size=1024,
     shuffle=True, pin_memory=True)
  test_loader = DataLoader(testset, batch_size=1024,
     shuffle=True, pin_memory=True)

  input_size = 784  # 28x28
  output_size = len(trainset.classes)
  model = RecognitionNet(input_size, output_size).to(device)
  loss_fn = nn.NLLLoss().to(device)
  optimizer = optim.SGD(model.parameters(), lr=lr, momentum=momentum)
```

Prikaz programskog koda 9 Pripremanje mreže na treniranje

Funkcija gubitka „NLLLoss“ (eng. *Negative log likelihood loss*) često se koristi pri radu sa klasifikacijama, a za izlaz ima negativnu vrijednost logaritma od predviđene klase. Na primjer, ukoliko tenzor upućuje da je učitana slika slovo A, funkcija gubitka uzeti će negativnu vrijednost logaritma vjerojatnosti da je na slici slovo A. Optimizator SGD ili stohastički gradijentalni pad (eng. *Stochastic gradient descent*) služi kako bi se dinamički mijenjale težine, sklonosti i stope učenja na što efektivniji način.

4.8 Petlja treniranja

```
for e in range(epoch_start, epochs):
      running_loss = 0
      for idx, (images, labels) in enumerate(train_loader):

        images = images.view(images.shape[0], -1)
        images = images.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()
        output = model(images)
        loss = loss_fn(output, labels.to(device))
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

        # Statistics
        predictions = output.argmax(dim=1, keepdim=True).squeeze()
        correct = (predictions == labels).sum().item()
        accuracy = correct / len(predictions)
```

Prikaz programskog koda 10 Petlja treniranja mreže

Treniranje uključuje prolazak kroz „train_loader“, pripremajući slike i oznake i slanje slika u mrežu. Može se primijetiti rad optimizatora te funkcije gubitka koji u pozadini mijenjaju varijable te mijenjaju vrijednosti unutar čvorova mreže. Na kraju se može uočiti način mjerenja preciznosti.

Petlja iz desetog programskog isječka iskorištena je više puta, a mreža je istrenirana mijenjajući broj epoha, stopu učenja, momentum, obradu slike i tako dalje. Navedeni podaci kao i točnost istreniranog modela biti će prikazana u tablici.

4.9 Petlja testiranja

```
num_correct = 0
num_samples = 0
model.eval()

# Bez mijenjanja gradijenata
with torch.no_grad():
  for images, labels in test_loader:
    images = images.view(images.shape[0], -1)
    images = images.to(device)
    labels = labels.to(device)

    out = model(images)
    _, preds = out.max(1)
    num_correct += (preds == labels).sum()
    num_samples += preds.size(0)

  print(f'Correct/total: {num_correct}/{num_samples}, { round(
    int(num_correct / num_samples * 100), 2) }%')
```

Prikaz programskog koda 11 Petlja testiranja istreniranog modela

Petlja u kojoj se procjenjuje preciznost istreniranog modela vrlo je slična petlji treniranja, no uz nekoliko pojednostavljenja. Na modelu je postavljen rad za provjeru, te su isključene automatske promjene gradijenata. Osim toga, izbačen je optimizator i funkcija gubitka.

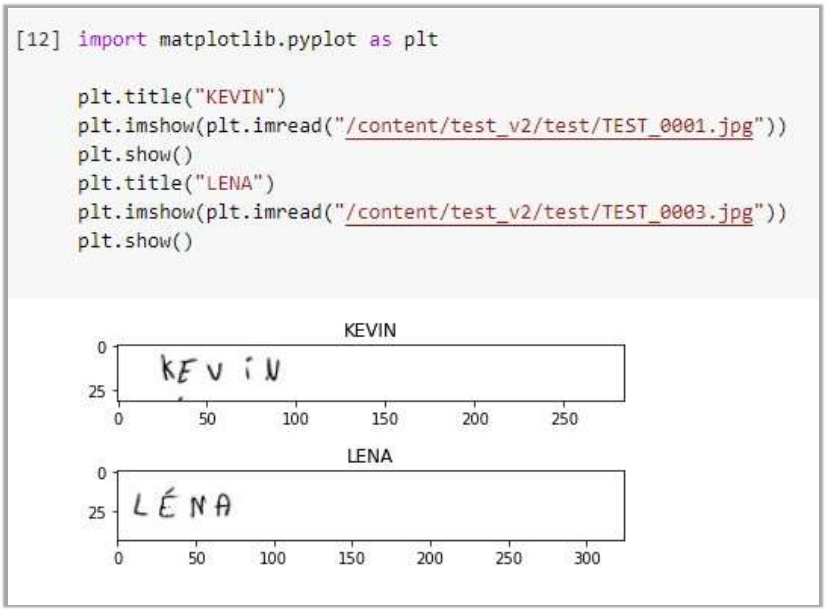
S obzirom na to da se u „DataLoader“ funkciju poslao „batch_size“ vrijednosti 1024, na izlazu modela dobije se niz od toliko cijelih brojeva koji označavaju indeks predviđene oznake. Usporedba „(preds == labels).sum()“ točno pogođene indekse postavlja na True, odnosno 1, a ostale na False ili 0, nakon čega se niz sumira. Na taj način usporedbom ukupnog broja i broja točnih predviđanja dolazi se do točnosti.

4.10 Pregled i analiza

Za analizu i olakšanje rada korišteno je nekoliko biblioteka i alata. Općenito u radu korišteno je mnogo njih, no u analizi i pregledu podataka, međurezultata i usporedbe točnosti najviše su pridonijeli Matplotlib, TensorBoard, pkbar i Pillow.

4.10.1 Matplotlib

U analizi i obradi izvornog podatkovnog skupa, ali i općenito u prikazu statističkih podataka i slika korišten je „matplotlib.pyplot“. Konkretno u završnom radu, pokazao se korisnim za traženje greške zbog koje je točnost predviđanja bila iznimno dobra prilikom treniranja, a loša prilikom testiranja mreže. Naime, otvaranjem jedne slike iz svakog direktorija, uočeno je da slike nisu obrađene na isti način i zbog toga mreža nije bila učinkovita.



Slika 14 Otvaranje slika bibliotekom Matplotlib

4.10.2 TensorBoard

TensorBoard je alat za vizualizaciju podataka. Izrađen je za TensorFlow, no može se koristiti i u radu sa PyTorch platformom. U sljedećim stavkama demonstrirano je pisanje i prikazivanje te analiziran izlaz.

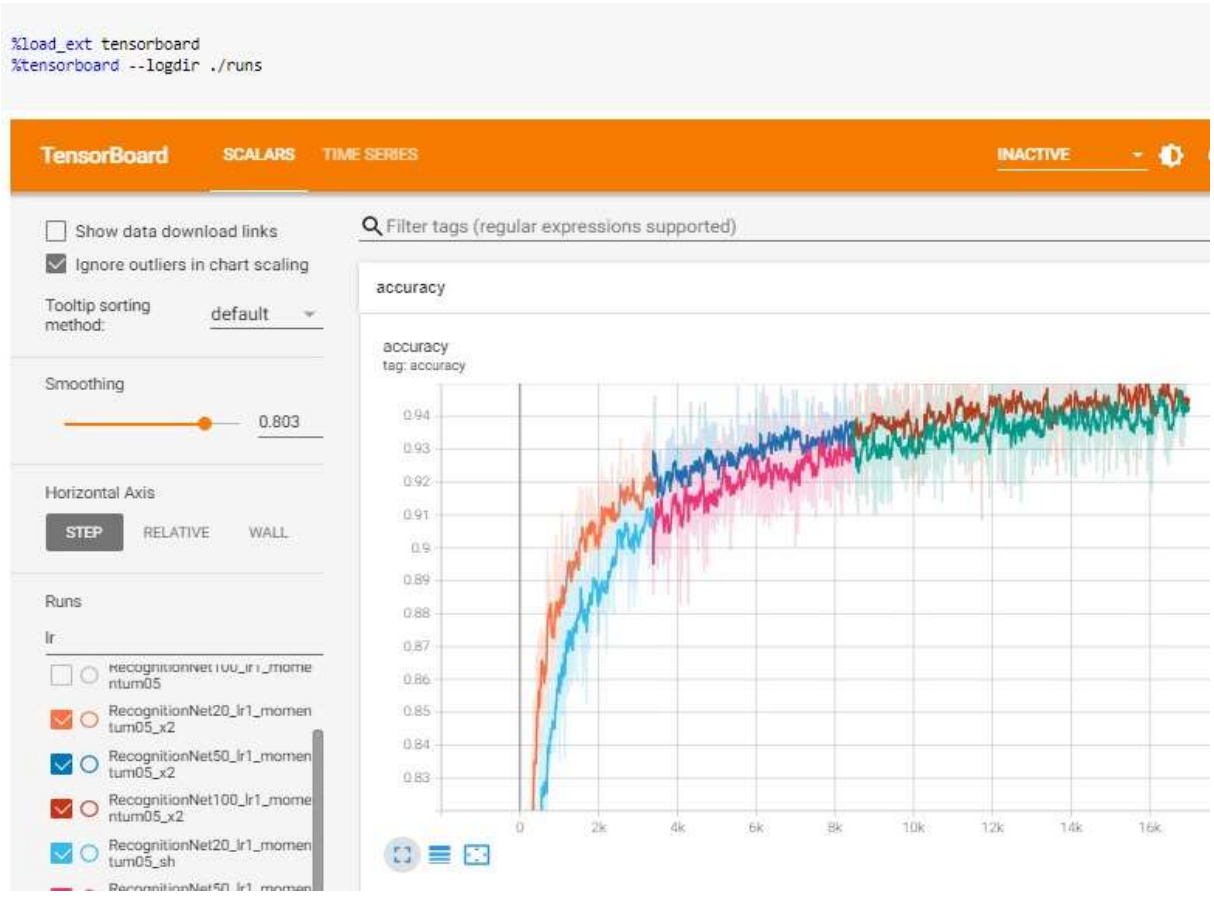
```python
# Import TensorBoard
from torch.utils.tensorboard import SummaryWriter

# Select data directory
writer = SummaryWriter(f'./runs/Model_2')

# Write data
writer.add_scalar('Accuracy (%)', 23.17, 10)
writer.add_scalar('Accuracy (%)', 32.78, 20)
#...

# Example in model training loop
writer.add_scalar('Loss', loss.item(), (epoch * items_per_epoch) + idx)
```

Prikaz programskog koda 12 Primjer TensorBoard rada



Slika 15 Primjer TensorBoard alata

4.11 Analiza rezultata

Neuronska mreža testirana je sa četiri različite konfiguracije, od kojih je testirana točnost nakon 20, 50 i 100 epoha. Najbolja točnost, blizu 93 posto, postignuta je uz manju stopu učenja od početne te bez obrade podataka prije prolaska.

Karakteristike \ Epoha	20	50	100
stopa učenja = 0.033 momentum = 0.5	86,71%	90,09%	91,75%
st. uč. = 0.1 mom. = 0.5 dupliciranje slika	90,53%	90,9%	90,51%
st. uč. = 0.1 mom. = 0.5 izoštrenje slika	89,07%	89,5%	90,25%
st. uč. = 0.1 mom. = 0.5	90,08%	92,16%	92,94%

Tablica 2 Konačni rezultati treniranja

5 ZAKLJUČAK

Kroz seminarski rad predstavljeno je područje strojnog učenja i umjetne inteligencije na više razina apstrakcije. Osim teorije potrebne za rad strojnog učenja i prepoznavanja teksta iz slika, analizirano je tržište i potencijal strojnog učenja u suvremenom svijetu. Osim toga, prikazani su i primjeri u kojima se ono koristi. Nadalje, opisana je izrada projekta aplikacije za prepoznavanje teksta iz slika kroz programski jezik Python. Kratkom analizom popularnih alata za izradu i detaljnijom obradom relevantne teorije obuhvaćeni su svi dijelovi kreiranja aplikacije, od rada sa tenzorima do treniranja podataka i predviđanja. Kao rezultat navedeni su dobiveni rezultati dizajnirane neuronske mreže, ali i rezultati suvremenih neuronskih mreža.

Popis literature

[1] Anderson, Dede, Fontana, Panikkar, Taylor and Waugh, World Book Encyclopedia, Chicago: World Book Inc., 2019.

[2] Ž. Panian, Informatički Enciklopedijski Rječnik (M-Z), Zagreb: Europapress holding d.o.o, 2005.

[3] C. M. B. Dana H. Ballard, Computer Vision, Englewood Cliffs: Prentice-Hall Inc., 1982.

[4] J. J. Hopfield, »Neural networks and physical systems with emergent collective computational abilities,« *Proceedings of the National Academy of Sciences of the United States of America,* pp. 2554-2558, Travanj 1982.

[5] »indeed,« [Mrežno]. Available: https://www.indeed.com/career/machine-learning-engineer/salaries?from=career.

[6] B. Hayer, »Business Broadway,« 8 Travanj 2018. [Mrežno]. Available: https://businessoverbroadway.com/2018/04/08/salaries-of-data-scientists-and-machine-learning-engineers-from-around-the-world/.

[7] »EL-PRO-CUS,« [Mrežno]. Available: https://www.elprocus.com/artificial-neural-networks-ann-and-their-types/.

[8] »Design and evolution of modular neural network architectures,« *Neural Networks, Svezak 7, Izdanje 6-7,* pp. 985-1004, 1994.

[9] P. Tahmasebi i H. Ardeshir, »Application of a Modular Feedforward Neural Network for Grade Estimation,« *Natural Resources Research,* pp. 25-32, 21 Siječanj 2011.

[10] F. Careers, »Facebook,« 15 Siječanj 2020. [Mrežno]. Available: https://www.facebook.com/careers/life/machine-learning-at-facebook.

[11] B. Shi, X. Bai i C. Yao, »An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition,« 21 Srpanj 2015. [Mrežno]. Available: https://arxiv.org/pdf/1507.05717.pdf.