TEHNIČKO VELEUČILIŠTE U ZAGREBU
STRUČNI STUDIJ RAČUNARSTVA

Stjepan Salopek

PREPOZNAVANJE TEKSTA RAČUNALNIM VIDOM

ZAVRŠNI RAD br. 1317

Zagreb, veljača, 2022.

TEHNIČKO VELEUČILIŠTE U ZAGREBU

STRUČNI STUDIJ RAČUNARSTVA

Stjepan Salopek

JMBAG: 0246084632

PREPOZNAVANJE TEKSTA RAČUNALNIM VIDOM

ZAVRŠNI RAD br. 1317

Zagreb, veljača, 2022.

Sažetak

Seminarski rad metodološki će prikazati izradu aplikacije prepoznavanja teksta računalnim vidom. Pojam računalnog vida obradit će se kroz tehnologiju umjetne inteligencije prolaskom kroz potencijal područja te tržišnu vrijednost. Obradit 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 navedeno znanje prikazat će u implementaciji programskog kôda i analizi konačnih rezultata točnosti.

Ključne riječi: strojno učenje, neuronske mreže, prepoznavanje teksta

SADRŽAJ

Popis tablica

Popis slika

Popis prikaza programskog kôda

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 velikom broju istih.

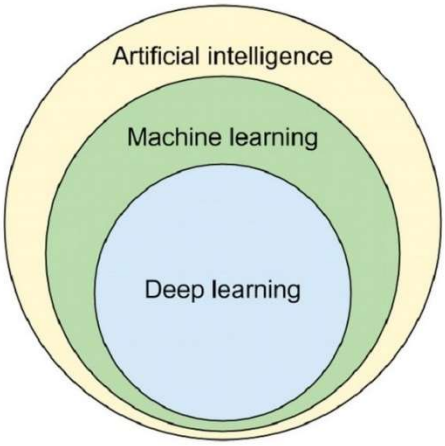
Nakon toga strojno učenje predstavlja se kao rješenje na probleme iz prakse koje čovjek teoretski može riješiti, no oduzima previše vremena, zbog čega pokušava poslove ranije nedostupne računalima učiniti efikasnijima. Nakon toga obrađuje se pojam neuronskih mreža i modela kao način implementacije strojnog 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 primjene strojnog učenja.

Nakon analize stanja tržišta, definiranja pojmova i uvoda u neuronske mreže, rad prelazi na praktični dio, na 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



Slika 1. Odnos umjetne inteligencije i strojnog učenja

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 koristit 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 ih i zatim prosljeđuju daljnjim neuronima, čime se tvori mreža [3] [4].

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 raširenо na tržištu kao što su mrežne aplikacije, 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].

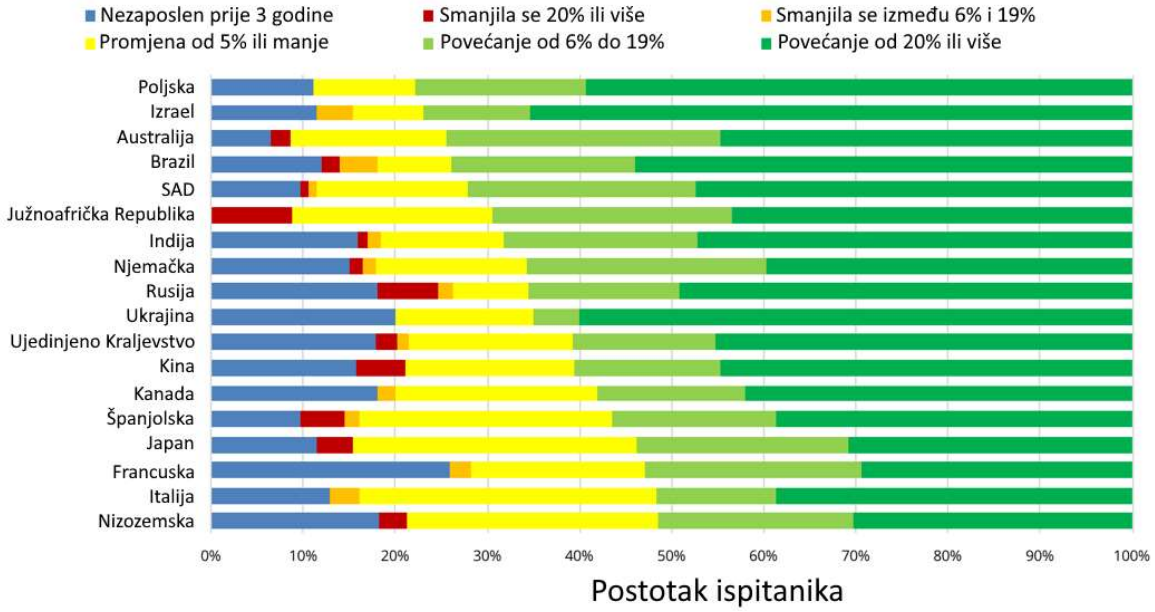
Država	**Godišnja plaća izražena u dolarima**
Sjedinjene Američke Države	120.000
Australija	110.719
Izrael	87.500
Kanada	81.330
Njemačka	80.120
Nizozemska	75.337
Japan	70.132
Ujedinjeno Kraljevstvo	66.209
Italija	59.791
Francuska	55.008
Južnoafrička Republika	50.051
Španjolska	47.833
Kina	41.310
Brazil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

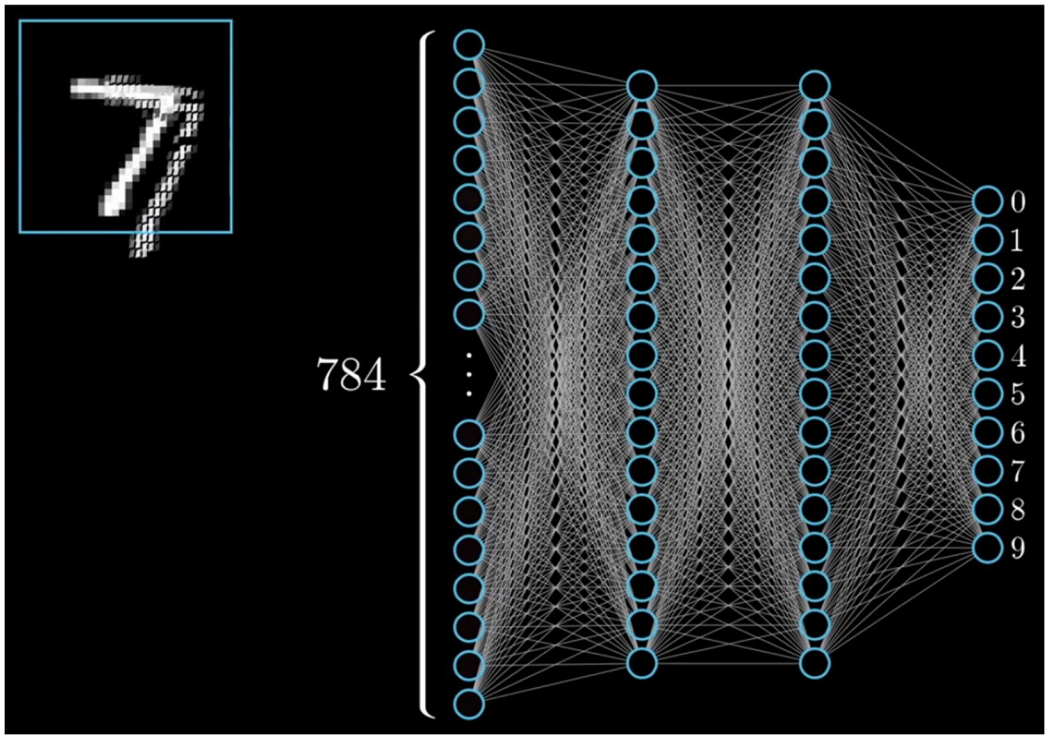
informacije? U jednostavnim slučajevima mogu doći od osoba, no zbog ograničenosti ljudske sposobnosti takvu vrstu informacije obično donosi računalo.

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 s 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 korisniku. Kao u mnogim situacijama, bilo bi idealno kada bi čovjek naporan i repetitivan posao mogao prepustiti računalu. 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 unijeti 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 tijekom 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

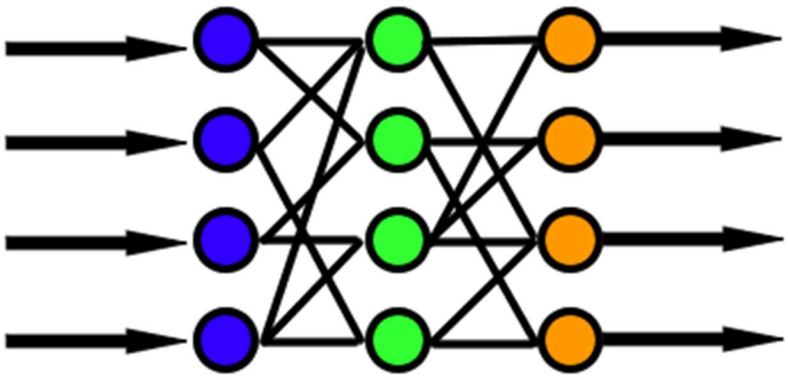
Neuronske mreže često se objašnjavaju na primjeru prepoznavanja brojeva. Koristeći navedene definicije neuronske mreže, može se uočiti da bi se mreža s treće slike koristila za prepoznavanje brojeva, da su joj ulazni parametri crno-bijele slike s različitim nijansama bijele i crne boje, da na ulazu ima 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 s apstraktnim informacijama poput linija, rubova, objekata i brojeva. Svrha učenja neuronske mreže jest izmijeniti funkcije koje određeni neuron obavlja tako da na se izlazu provjeri 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 se moraju 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 koristit će se engleski nazivi modela neuronskih mreža radi očuvanja semantičkog značenja.

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 (engl. *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 s digitalnom grafikom pri detekciji, klasifikaciji i segmentaciji objekata.



Slika 4. Primjer jednostavne feedforward mreže

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.

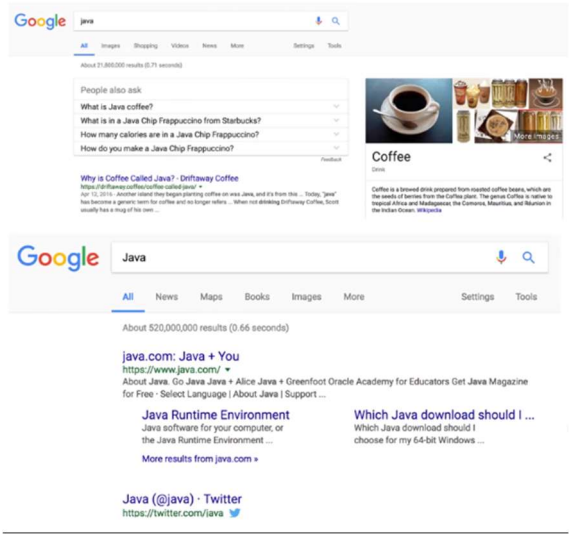
Recurrent neuronske mreže slične su prvoj navedenoj vrsti, no razlika je što u njima informacije mogu putovati u oba smjera, odnosno isti neuron i njegova memorija može se koristiti za obradu veće količine podataka. Zbog tog svojstva obično se primjenjuju u primjeni prepoznavanja rukopisa ili prepoznavanja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



Slika 6. Strojno učenje u Google tražilici

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 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.

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 uštedjeti tako što će tražilica pokušati predvidjeti koje znanstvene radove ili članke on traži.

3. STVARANJE APLIKACIJE

3.1. Odabir tehnologija

U sljedećim odlomcima obradit 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 (engl. *Jupyter Notebooks*) su razvojna okruženja smještena na javnom ili lokalnom serveru koja omogućuju pisanje Python kôda, naredbi komandne linije (engl. *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

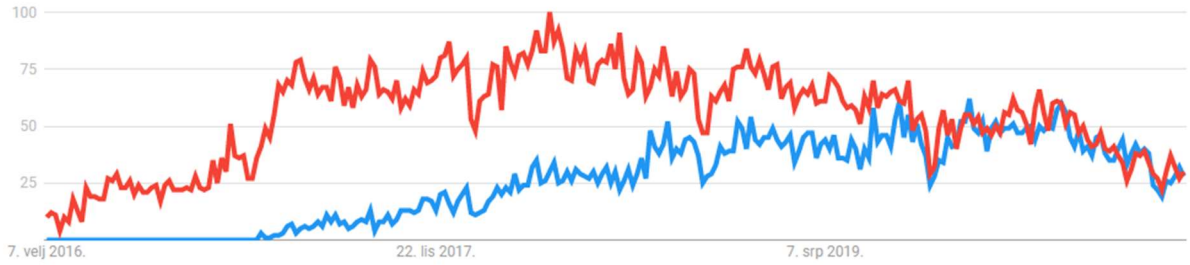
TensorFlow je paket otvorenog kôda namijenjen za numeričke izračune visokih performansi korištenjem grafova protoka podataka (engl. *Data flow graph*). U pozadini TensorFlow je radni okvir koji izvodi računske operacije tenzorima. Prednosti uključuju mnoge algoritme unutar paketa, ogromnu zajednicu,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

rada kroz više grafičkih kartica i procesora. Nedostaci su niža brzina izvođenja u odnosu na druge biblioteke, ali i težina učenja za početnike.

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 na Googleovoj tražilici. Može se uočiti da je TensorFlow, označen crvenom bojom, dugo vremena bio popularniji na tržištu, no da je u zadnje vrijeme otprilike jednako popularan kao PyTorch.



Slika 7. Vremenski prikaz pretraživanja riječi *tensorflow* i *pytorch*

3.2. Teorija i funkcionalnosti potrebni za izradu

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

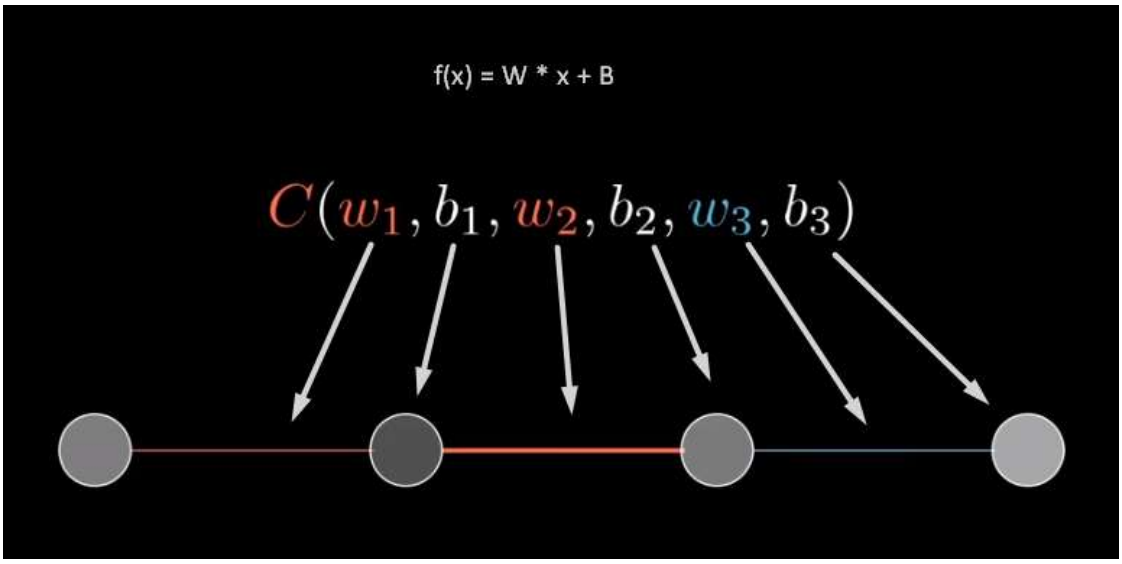
3.2.1. Tenzori

Tenzor je izraz posuđen iz matematike, a u strojnom učenju označava strukturu sličnu matricama višeg reda. Tenzori u PyTorchu sastoje se 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, ovisno o potrebama. Neuroni su povezani standardnim linearnim funkcijama oblika $f(x) = a * x + b$.

Parametar „a“ linearne funkcije naziva se težina (engl. *weight*), parametar „b“ označava sklonost (engl. *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.



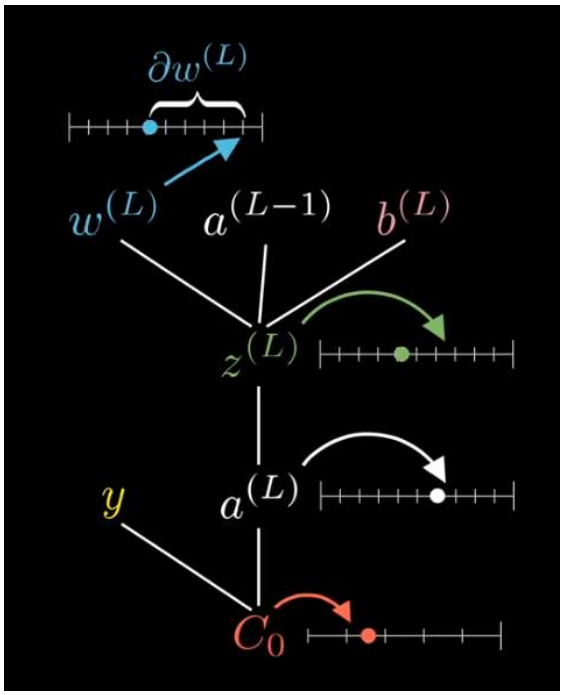
Slika 8. Propagacija neuronske mreže

Na osmoj slici može se vidjeti prikaz jednostavne neuronske mreže s 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. Propagacija unazad

Uobičajena propagacija koristi se za predviđanje kada se zna koje težine i sklonosti treba postaviti na čvorove dizajnirane mreže. Na primjer, za mrežu s dva čvora u kojoj je mreža istrenirana da izlaz bude vrijednost 2.5, iz formule $f(x) = W * x + B = 2.5$ može se 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 se žele predvidjeti kompleksnije stvari, na primjer nalazi li se na slici mačka ili pas, tada se koristi metoda propagacije unazad. Slično kao što se u prethodnom odlomku mreža istrenirala za izlaz 2.5 postavljajući težinu i sklonost, tako se propagacija unazad koristi da se pokuša predvidjeti vrijednost težina i sklonosti u mnogo većim mrežama, i time dobije željeni izlaz.



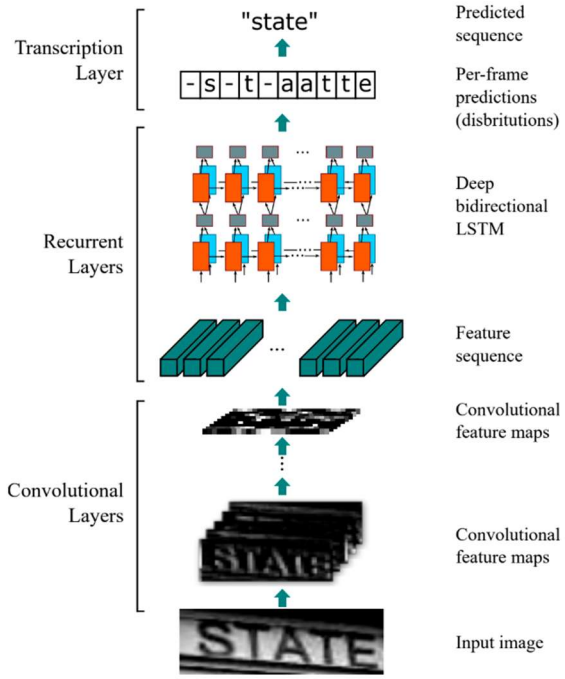
Slika 9. Ovisnost neurona u neuronskoj mreži

Na devetoj 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 bi se dobila ovisnost izlazne vrijednosti o ulaznoj, mora se derivirati funkciju $f(x)$, čime se dobije gradijent tog čvora. Tada, ukoliko je gradijent malen, male promjene varijable „x“ rezultirat 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 propagacije unazad kroz cijelu mrežu paket PyTorch automatski sprema derivacije ili gradijente čvorova u njih same, te se na temelju toga može dobiti preciznija izlazna vrijednost.

Propagacija unazad 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. Skica modela prepoznavanja 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 rezultiraju predviđanjem 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 definiranje propagacije unazad 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 modelu neuronske mreže pošalje objekt, primjerice slika koja sadrži tekst.

U neuronskim mrežama, pojam *max pooling* označava smanjivanje veličine slike metodom uzimanja najveće vrijednosti piksela iz odabrane matrice. Na primjer, prolaskom matrice dimenzija 2x2 kroz sliku dimenzija 28x28, uzimajući maksimalne vrijednosti svakog piksela matrice, u konačnici bi se dobila slika dimenzija 14x14. Na ovaj način ubrzava se proces učenja smanjenjem ulaza.

Padajući sloj (engl. Dropout layer) označava uzimanje skupa vrijednosti, i poništavanjem nasumičnih članova. Primjerice, da se na niz od sto vrijednosti primjerni padajući sloj vjerojatnosti 50%, tada bi otprilike pedeset vrijednosti

imale vrijednost nula, dok bi ostale bile netaknute. Ova metoda često se koristi za sprječavanje pretreniranja dubokih mreža.

3.3. Rad s 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 Stohastički gradijentalni pad (engl. *Stochastic Gradient Descent*), ili SGD funkcija. Navedena funkcija težinu mijenja pomoću njene trenutne vrijednosti, gradijenta dobivenog u propagaciji unazad i konstante poslane koja određuje brzinu promjene:

$$weight = weight - learningRate \times gradient$$

```
for ponavljanje in range(2):
    for indeks, 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 s tekstom i slanjem tih podataka u objekt ranije kreirane koja predstavlja neuronsku mrežu. Proces se odvija više 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 trećeg i četvrtog programskog isječka može se primijetiti sličnost. Kada je u pitanju predviđanje, prolazi se kroz petlju šaljući slike kroz mrežu. Da bi se međurezultati treniranja mogli analizirati, koriste se slike čiji se tekst unaprijed zna. Prilikom računanja točnosti u četvrtom programskom kodu, 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 s 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

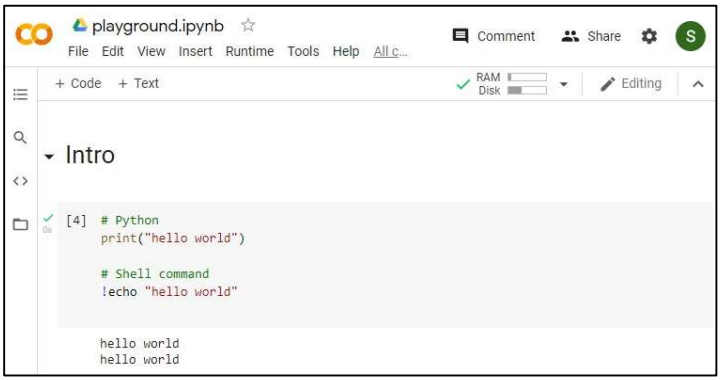
4.1. Plan rada

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 s 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. Istrenirat će se više modela i za svaki testirati preciznost čitanja slova.

4.2. Postavljanje razvojnog okruženja

Kompletan praktični dio rada odrađen je na Googleovoj platformi. Nakon dovršetka, kod iz Jupyter bilježnice pretvoren je u uobičajenu strukturu Python projekta. Google Drive korišten je za pohranu podataka kao što su grafovi i istrenirani modeli, a Colaboratory ili Colab za obradu.



Slika 11. Rad u Colaborytoryju

Colab pojednostavljuje rad jer ima funkcionalnosti poput uvoza Google Drivea direktno u Linux virtualnu mašinu na kojoj je pokrenuta Jupyter bilježnica. Također, u bilježnici se mogu direktno pokretati naredbe ljuske što se pokazalo korisnim u analizi podataka TensorBoard alatom. Iako Colab omogućuje pristup grafičkim karticama, vremenski su ograničene. Stoga se treniranje odrađivalo u više navrata s vremenskim razmakom od jedan do dva dana.

4.3. Podatkovni skup

Prema Oxfordu, podatkovni skup je skup podataka koji računalo promatra kao logičku cjelinu [12]. U kontekstu strojnog učenja, podatkovni skup označava skup brojčanih, tekstualnih, vizualnih ili auditivnih podataka. Primjer tekstualnog podatkovnog skupa je popis putnika Titanika. Na temelju popisa, neuronska mreža može napraviti procjenu šanse preživljavanja na temelju ulaznih parametara kao što su dob, spol ili klasa osobe.

4.5. MNIST i EMNIST skupovi

Jedan od najpopularnijih vizualnih podatkovnih skupova je MNIST skup. Smatra se školskim primjerom skupa zbog svoje jednostavnosti te se često koristi kada se žele testirati druge tehnologije unutar strojnog učenja, a kada podatkovni skup nije od velike važnosti. Sastoji se od 70.000 slika veličine 28x28 piksela širine 1 bita, a podijeljene su u deset klasa, na znamenke od 1 do 9.

U radu je korištena varijacija navedenog skupa koji se naziva EMNIST (*engl. Extended MNIST*). Sastoji se od 300.000 slika, od čega je 80% korišteno za treniranje neuronske mreže, a 20% za evaluaciju iste. Veličina crno-bijelih slika također je 28x28 piksela, a podijeljene su u dvadeset šest klasa, po jedna za svako slovo engleske abecede.

4.5. Dizajniranje neuronske mreže

Problem prepoznavanja teksta riješen je pomoću šest originalnih neuronskih mreža. Prve dvije mreže dizajnirane su korištenjem teorije, a ostale četiri su dizajnirane spajanjem raznih karakteristika postojećih mreža koje su često korištene za druge podatkovne skupove.

Mreže su nazvane prema originalnim nazivima ili okvirnim karakteristikama na engleskom jeziku. Prve dvije su naziva ConvNet i ResidualNet. Slijede MnistSimpleCNN, SimpleNetv1, Resnet18 te modificirani Resnet18.

4.5.1. ConvNet

ConvNet naziv je za malenu konvolucijsku mrežu koja se sastoji od dvije vrste blokova: konvolucijskog i linearnog bloka. Propagacija slike kroz mrežu sastoji se od prolaska kroz sedam blokova. Prvih pet blokova su konvolucijski i na sebi sadrže padajući (engl. dropout) sloj. Prva dva bloka dodatno primjenjuju i *max pooling*. Nakon toga vrijednosti se pretvaraju iz višedimenzionalnih matrica koje stvara konvolucija do jednodimenzionalnog niza vrijednosti. Niz se nadalje šalje kroz dva linearna bloka koji u konačnici za izlaz imaju 26 vrijednosti.

Konvolucijski blok sastoji se, redom, od konvolucije matrice veličine 3x3, normalizacijskog sloja, opcionalnog *max pooling* sloja te padajućeg sloja. Linearni blok sličan je konvolucijskom, no umjesto konvolucije sadrži linearni sloj, te ne vrši *max pooling*. Obje vrste blokova nakon normalizacije primjenjuju ReLU aktivacijsku funkciju.

Mreža je trenirana sveukupno 80 epoha dostižući točnost od 94.68%. Najbolju točnost od 94.86% postigla je u 49. epohi.

4.5.2. ResidualNet

ResidualNet je varijacija konvolucijske neuronske mreže koja primjenjuje metodu ostatka (engl. residue). Metoda ostatka na određenom mjestu tijekom prolaska slike kroz mrežu sprema vrijednosti, prolazi dalje kroz mrežu, i zbraja ih s novom vrijednošću kasnije. Na taj način mreža, slikovito rečeno, uči nove obrasce u podacima uzimajući u obzir one ranije naučene. Navedena metoda često se koristi za sprječavanje pretreniranja.

Mreža se sastoji od tri vrste bloka: rezidualnog, konvolucijskog i linearnog. Prolazak kroz jedan rezidualni blok uključuje spremanje ulaza, prolazak kroz dva konvolucijska bloka te vraćanje zbroja konvolucijskog izlaza s inicijalnom ulaznom vrijednošću.

Kompletni prolazak kroz ResidualNet mrežu vrši se na sljedeći način. Slika prolazi kroz dva konvolucijska bloka koja primjenjuju *max pooling*, nakon čega slijede tri rezidualna bloka te nakon njih dva linearna.

Mreža je trenirana sveukupno 86 epoha dostižući točnost od 94.54%. Najbolju točnost od 94.97% postigla je u 50. epohi.

4.5.3. MnistSimpleCNN

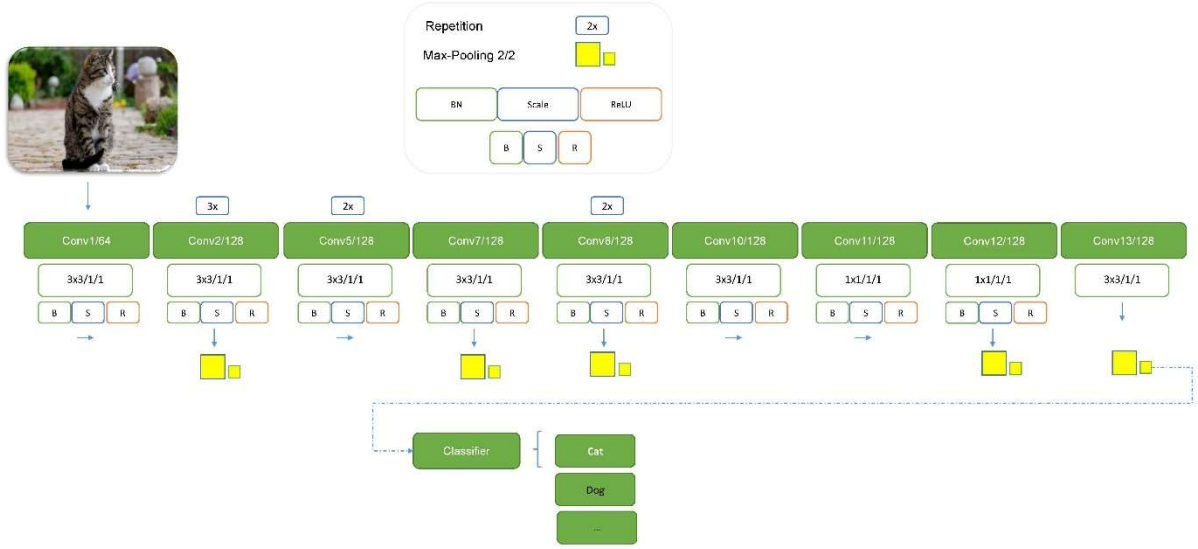
MnistSimpleCNN zajednički je naziv za tri vrste konvolucijskih neuronskih mreža [13]. Inicijalno su dizajnirane za MNIST podatkovni skup, ali su prilagođene EMNIST skupu. Prva vrsta ove mreže sastoji se od deset konvolucijskih blokova matrice veličine 3x3, nakon čega slijedi jedan linearni sloj. Druga vrsta ima pet konvolucijskih blokova veličine 5x5, dok treća ima četiri bloka veličine 7x7. Sve mreže nakon linearnog sloja primjenjuju normalizaciju podataka.

Mreža matrice 3x3 trenirana je na više načina, a maksimalno 40 epoha dajući točnost 94.15%. Najveću točnost od 95.01% imala je nakon 12 epoha.

Mreža matrice 5x5 trenirana je 40 epoha s izlaznom točnošću od 94.09%, a najbolju točnost od 94.86% postigla je u 7. epohi.

Mreža matrice 7x7 trenirana je 30 epoha s točnošću od 94.19%, a najbolju točnost od 94.67% imala je nakon 13 epoha.

4.5.4. SimpleNetv1

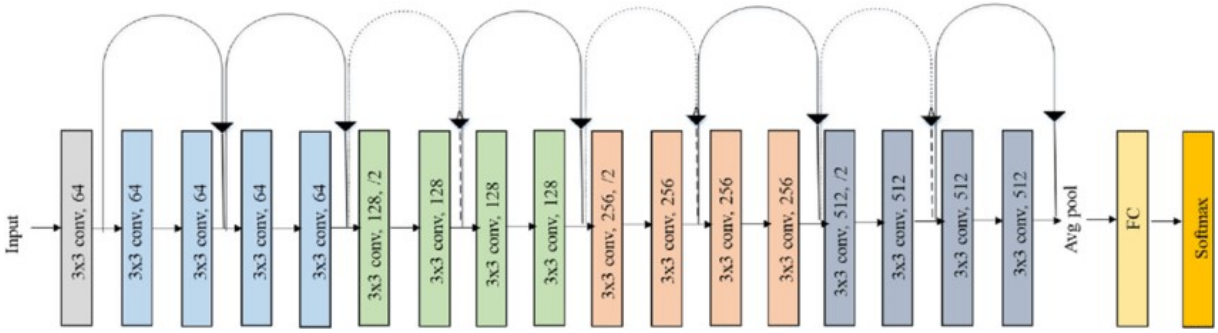


Slika 12. Prikaz SimpleNetv1 neuronske mreže

Mreža SimpleNetv1 konvolucijska je mreža izrađena za popularne CIFAR10, CIFAR100 i MNIST podatkovne skupove [14]. Prilagođena je EMNIST skupu. Sastoji se od sveukupno trinaest konvolucijskih blokova, nakon čega slijedi linearni sloj naziva klasifikator. Prema dvanaestoj slici koja definira strukturu mreže, može se primijetiti da se u većini slučajeva koristi konvolucijski blok matrice 3x3, dok varijacije sadrže matricu 1x1, a na nekoliko mjesta vrši se *max pooling*.

Mreža je trenirana na više načina, maksimalno 45 epoha s točnošću od 93.85%. Najvećom točnošću od 94.51% rezultirala je 14. epoha.

4.5.5 Resnet18



Slika 13. Resnet18 neuronska mreža

Resnet je popularna duboka konvolucijska neuronska mreža koja primjenjuje tehniku ostatka (engl. residue) [15]. Inicijalno dizajnirana za ImageNet podatkovni skup, prilagođena je EMNIST skupu. Kao što se može vidjeti na trinaestoj slici, sastoji se od sedamnaest konvolucijskih blokova, koji zbrajaju ostatke na svakom drugom bloku, nakon čega se vrši *pooling* metodom prosječne vrijednosti. Nadalje, konačni izlaz dobiva prolaskom kroz linearni sloj i primjenom Softmax funkcije.

Mreža je trenirana na više načina, a najveća, 40. epoha dala je točnost od 94.29%. Najveća točnost postignuta je u 15. epohi, a iznosi 94.63%.

4.5.6. Modificirani Resnet18

Istraživanjem je pronađen znanstveni rad koji primjenjuje dubinsko padajuću metodu na rezidualne mreže [16]. Navedena metoda prije zbrajanja ostatka i izlaza konvolucije, na ostatak se primjenjuje padajući sloj vjerojatnosti 25, 50 ili 75%. Motiv znanstvenog rada bio je smanjiti vrijeme treniranja duboke neuronske mreže Resnet18, no uz smanjeno vrijeme treniranja zamijećene su, pod određenim uvjetima, veće točnosti neuronske mreže. Neuronska mreža je u praktičnom radu nazvana Resnet18_dd, gdje dd označava dubinsko padajuću metodu (engl. Depth dropout).

Mreža se pokazala kao najtočnija među ostalima, te je iz tog razloga provedeno najviše napora da bi se prilagodili parametri za što veću točnost. Od nekoliko raznih metoda treniranja, maksimalna istrenirana epoha je 42. uz točnost od 95.55%, dok je maksimalna točnost od 95.59% postignuta u 30. epohi.

Glavna karakteristika Resnet18_dd mreže je otpornost na pretreniranje. Imajući na umu veliku vremensku razliku u treniranju između 42. i 30. epohe, obzirom na kompleksnost mreže, malena razlika u točnosti potiče na taj zaključak.

4.6. Učitavanje podataka

Pri radu s podacima PyTorch pruža mnoge mogućnosti preko biblioteka. Za učitavanje kompletnog podatkovnog skupa korištena je funkcija EMNIST koja se nalazi u imenovanom području torch.datasets.

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])
trainset = torch.datasets.EMNIST('data', split='letters', train=True,
                download=True, transform=transform)
testset = torch.datasets.EMNIST('data', split='letters', train=False,
                download=True, transform=transform)
```

Prikaz programskog koda 5. Učitavanje i transformacija slika

U programskom isječku mogu se primijetiti i transformacije podataka iz biblioteke „torchvision.transforms“ koje su zadužene za pripremanje slika za rad. Funkcija transforms.Resize korištena je u slučajevima kada je na ulazu očekivana slika dimenzija 224x224. Funkcija transforms.Normalize korištena je da bi se binarni zapisi slika izgladili, čime se postigla veća točnost.

4.7. Postavljanje varijabli

Prilikom rada aplikacije koriste se mnoge promjene varijabli koje uvelike utječu na rad. Objekti „DataLoader“ klase, vidljivi na desetom programskom isječku, zaslužni su za miješanje slika te za višestruko učitavanje istih. Varijabla koja određuje broj istovremeno obrađenih slika naziva se batch_size. Osim DataLoader objekata, potrebno je stvoriti i objekt neuronske mreže, objekt funkcije gubitka te optimizator. Nadalje, 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 „tqdm“ biblioteka koja prikazuje korisne vrijednosti tijekom treniranja.

```
e=0 curr=0:  60%|██████    | 586/975 [07:58<05:16, 1.23it/s, acc=57.8]
```

Prikaz programskog koda 6. Prikaz tqdm statistike

```
train_loader = DataLoader(trainset, batch_size=256,
     shuffle=True, pin_memory=True)
test_loader = DataLoader(testset, batch_size=32,
     shuffle=True, pin_memory=True)


model = ResidualNet(output_size=27)
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=1e-3, momentum=0.9)
```

Prikaz programskog koda 7. Pripremanje mreže na treniranje

Funkcija gubitka *CrossEntropyLoss*, često se koristi pri radu s klasifikacijama, a za izlaz ima realne vjerojatnosti da se na slici nalazi klasa, između 0 i 1, što označava postotke od 0 do 100. Na primjer, ukoliko je učitana slika slova A, funkcija gubitka istrenirane mreže će na indeksu klase A postaviti najveću vrijednost.

Optimizator SGD ili stohastički gradijentalni pad (engl. *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(0, epochs):
      for idx, (images, labels) in enumerate(train_loader):

        optimizer.zero_grad()
        output = model(images)
        loss = loss_fn(output, labels)
        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 8. Petlja treniranja mreže

Treniranje uključuje prolazak kroz objekt train_loader i slanje slika u mrežu, koje se vrši više puta, ovisno o ukupnom broju epoha. Može se primijetiti rad optimizatora i funkcije gubitka koji u pozadini mijenjaju varijable i vrijednosti unutar čvorova mreže. Na kraju se može uočiti način mjerenja preciznosti.

Na taj način vrši se jedno testiranje. Provedeno ih je mnogo, a mijenjali su se razni podaci kao što je neuronska mreža, optimizator, oblik slika EMNIST podatkovnog skupa kao i korištenje lr_scheduler objekta u određenim slučajevima pomoću kojeg se planski smanjivala stopa učenja.

4.9. Petlja testiranja

```
total, correct = 0, 0
model.eval()

with torch.no_grad():
  for images, labels in test_loader:
    out = model(images)
    _, preds = out.max(1)
    num_correct += (preds == labels).sum()
    num_samples += preds.size(0)

  accuracy = num_correct / num_samples * 100
  print(f'Accuracy: { round(accuracy, 2) }%')
```

Prikaz programskog koda 9. Petlja testiranja istreniranog modela

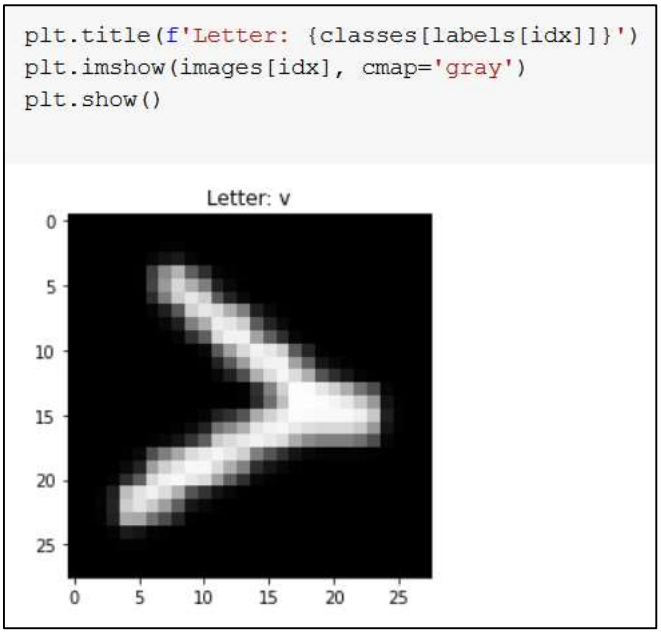
Petlja u kojoj se procjenjuje preciznost istreniranog modela vrlo je slična petlji treniranja, no uz nekoliko pojednostavljenja. Na modelu je postavljen evaluacijski način te su isključene automatske promjene gradijenata da bi se ubrzala procjena. Osim toga, u procjenjivanju točnosti nisu potrebni optimizator i funkcija gubitka.

U DataLoader funkciju poslao se batch_size veličine 256 prilikom treniranja za smanjenje vremena treniranja i 32 prilikom testiranja radi povećanja točnosti. Zbog toga se na izlazu modela dobije se niz od 256 nizova duljine 26 realnih brojeva, po jedan za svaku klasu, koji označavaju vjerojatnost klase na slici. Izlazni niz šalje se u metodu max(1) koja vraća niz od 256 cijelih brojeva u spremnik preds, označavajući indeks najvjerojatnijeg slova određene slike. Usporedba (preds==labels).sum() točno pogođene indekse postavlja na True, odnosno 1, a ostale na False ili 0, nakon čega se niz sumira. Suma označava ukupan broj točnih vrijednosti. Na taj način usporedbom ukupnog broja i broja točnih predviđanja dolazi se do točnosti.

4.10. Pregled i analiza

Za analizu je korišteno nekoliko biblioteka i alata. U analizi i pregledu podataka, međurezultata i usporedbe točnosti najviše su pridonijeli Matplotlib, TensorBoard i tqdm .

4.10.1. Matplotlib



Slika 14. Prikaz slike matplotlib bibliotekom

U analizi i obradi podatkovnog skupa, ali i općenito u prikazu statističkih podataka i slika korišten je „matplotlib.pyplot“. Konkretno u završnom radu, pokazao se korisnim za analizu slika i primjenjivanje transformacija na slikama.

4.10.2. TensorBoard

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

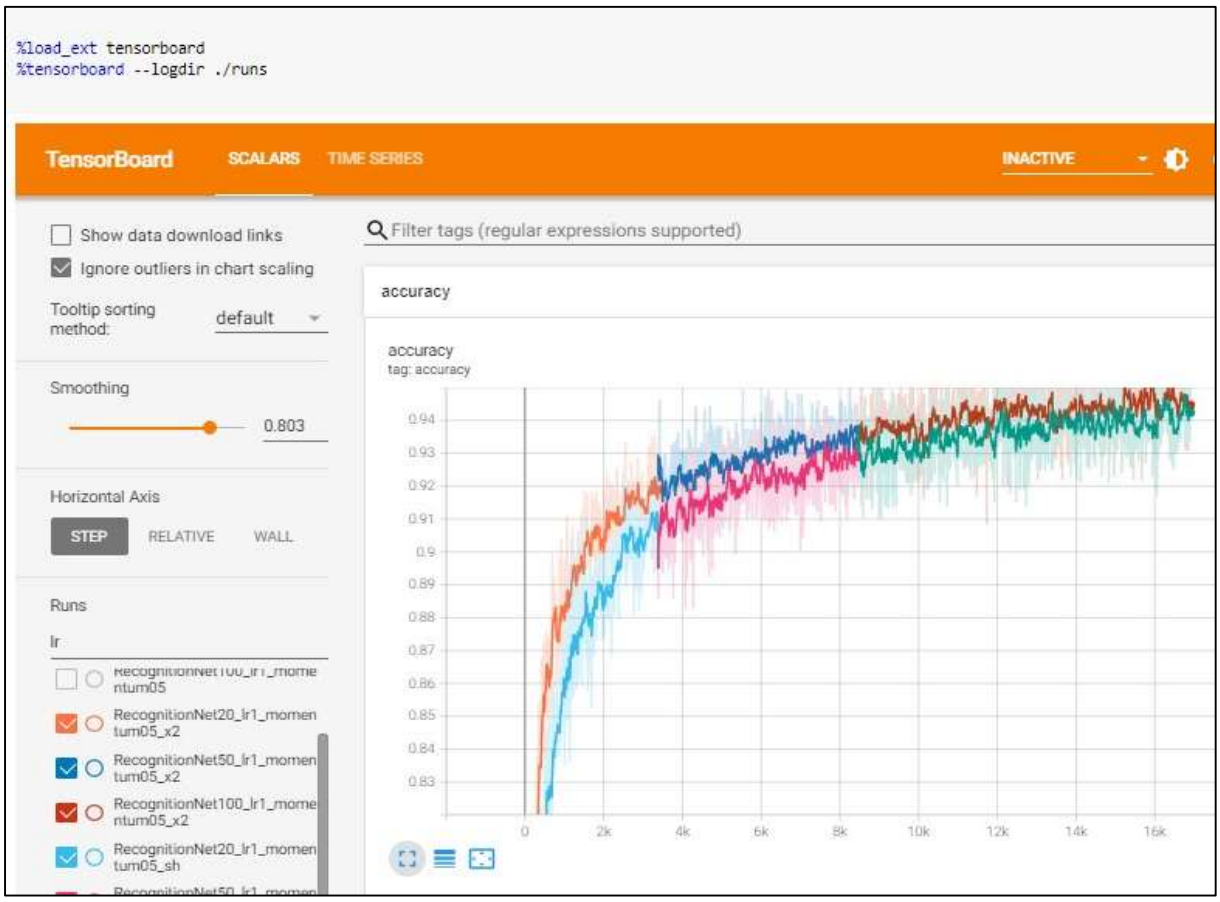
```python
# Import TensorBoard
from torch.utils.tensorboard import SummaryWriter

# 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 10. Primjer TensorBoard rada



Slika 15. Primjer TensorBoard alata

4.11. Tablica mjerenja

Mreža	Epoha poslj.	Točnost poslj.	Epoha najbolja	Točnost najb.
Resnet18_dd 25% SGD, lr_sched, 224x224 img	42	95.55%	30	95.60%
MnistSimpleCNN 3x3 Adam	29	94.56%	12	95.01%
ResidualNet Adam	86	94.54%	50	94.97%
MnistSimpleCNN 7x7 Adam	89	94.10%	12	94.96%
Resnet18_dd 25% SGD, lr_sched	19	94.80%	18	94.88%
ConvNet Adam	79	94.68%	49	94.86%
MnistSimpleCNN 5x5 Adam	29	94.35%	7	94.86%
Resnet18_dd 25% Adam, 224x224 img	29	94.62%	17	94.86%
Resnet18_dd 50% Adam, 224x224 img	29	94.39%	18	94.81%
Resnet18_dd 75% Adam	39	94.50%	14	94.78%
Resnet18_dd 75% Adam, 224x224 img	26	94.54%	12	94.78%
Resnet18_dd 50% Adam	29	94.69%	8	94.73%
Resnet18_dd 25% Adam	39	94.32%	14	94.72%
MnistSimpleCNN 3x3 SGD, lr_sched	29	94.42%	10	94.70%
MnistSimpleCNN 7x7 Adam, dropout	29	94.19%	13	94.67%
Resnet18 Adam	39	94.29%	15	94.63%
Resnet18_dd 75% SGD, lr_sched	14	94.55%	12	94.63%
Resnet18 Adam, 224x224 img	29	94.58%	13	94.59%
Resnet18_dd 25% SGD, lr_sched, 224x224 img, 10% dropout	24	94.57%	24	94.57%
MnistSimpleCNN 3x3 Adam, dropout	39	94.15%	19	94.54%
MnistSimpleCNN 5x5 Adam, dropout	39	94.09%	19	94.52%

Tablica 2. Mjerenja sortirana padajući po najboljoj točnosti

4.12. Definiranje parametara mjerenja

Druga tablica pokazuje rezultate svih mreža koje su imale najbolju točnost veću od 94%. Treniranje mreža vršilo se raznim metodama. Prvi stupac prikazuje opis modela, drugi i treći prikazuju epohu i točnost posljednjeg treniranog modela, dok četvrti i peti stupac prikazuju epohu i točnost najpreciznijeg modela. Ćelije prvog stupca u prvom redu sadrže naziv mreže modela, dok ostali redovi sadrže njegove karakteristike.

Korišteni optimizatori su Adam, koji automatski mijenja stopu učenja, te SGD, kojemu je objekt naziva lr_scheduler svakih deset epoha smanjio stopu učenja za koeficijent 10. Slike na ulazu bile su crno-bijele i dimenzija 28x28 osim na modelima gdje je naznačena dimenzija 224x224. Karakteristika dropout označuje primjenu padajućeg sloja vjerojatnosti 10% na izlaz konvolucije. Neovisno o tom sloju, na modele neuronske mreže Resnet18_dd primjenjivao se padajući sloj na ostatak prije zbrajanja s izlazom konvolucije.

4.13. Analiza mjerenja

Većina mreža približila se točnosti od 95%. S točnošću od 95.6%, model mreže Resnet18_dd je za više od pola posto premašio ostale. Razlog tomu je dubina mreže i kombinacija SGD optimizatora s mijenjajućom stopom učenja.

4.13.1. Optimizatori i pretreniranje

U početnim treniranjima modela, korišten je isključivo Adam optimizator koji se pokazao najjednostavnijim za korištenje zbog automatskog postavljanja stope učenja. Međutim, iako je brzo dostizao visoke stope učenja, nakon određene točke ona je počela opadati i nastalo je pretreniranje.

Iz tog razloga, nekoliko najboljih modela trenirano je iznova sa SGD optimizatorom i korištenjem objekta naziva lr_scheduler. Početna vrijednost stope učenja postavljena je na 10^{-3}, a svakih deset epoha smanjena je za koeficijent 10. Ova metoda pokazala se sporijom, ali robusnijom u konvergiranju prema najboljoj potencijalnoj točnosti mreže.

Problem pretreniranja često ograniči napredak neuronskih mreža. U praksi se može uočiti stagnacijom ili opadanjem točnosti s vremenom. U nekolicini modela problem se pojavio, što se može zaključiti iz razlike posljednje i najbolje točnosti. Najpreciznija mreža ujedno je i najotpornija na pretreniranje s razlikom

točnosti od 0.05%. Razlike u manje otpornim mrežama iznose do 0.5% blokirajući daljnji napredak mreže.

Rješenje problemu, što je također vidljivo u tablici, je korištenje metode padajućeg sloja, kao i metode ostatka. Mreža Resnet_dd sadrži obje metode te iz tog razloga očekivano je vidjeti mnoge njene varijacije na tablici najboljih modela.

5. ZAKLJUČAK

Kroz seminarski rad predstavljeno je područje strojnog učenja kroz teoriju i praktični rad. Uz teoriju potrebnu za prepoznavanje teksta iz slika, analizirano je tržište i potencijal strojnog učenja u suvremenom svijetu. Prikazani su i primjeri u kojima se ono koristi.

Nadalje, opisana je izrada projekta aplikacije za prepoznavanje teksta iz slika kroz programski jezik Python.Kroz izradu praktičnog rada obuhvaćeni su svi dijelovi izrađivanja projekta, od razvojnog okruženja i slika korištenih za prepoznavanje, do logike iza kreiranja modela neuronske mreže. Navedeni su svi korišteni modeli i njihove varijacije. Prikazani su rezultati mjerenjâ istreniranih modela i izvedeni zaključci na temelju rezultata. Dostigla se najviša točnost prepoznavanja od 95,6%.

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.

[12] »Oxford - Data set,« [Mrežno]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/data-set?q=data+set.

[13] S. An, M. Lee, S. Park, H. Yang i J. So, »arXiv - MnistSimpleCNN,« 5 8 2020. [Mrežno]. Available: https://arxiv.org/pdf/2008.10400v2.pdf.

[14] S. H. Hasanpour, M. Rouhani, M. Fayyaz i M. Sabokrou, »arXiv - SimpleNet,« 14 2 2018. [Mrežno]. Available: https://arxiv.org/pdf/1608.06037v7.pdf.

[15] K. He, X. Zhang, S. Ren i J. Sun, »arXiv - Deep Residual Learning for Image Recognition,« 10 12 2015. [Mrežno]. Available: https://arxiv.org/pdf/1512.03385.pdf.

[16] J. Guo i S. Gould, »AU National Universite - Depth Dropout: Efficient Training of Residual Convolutional Neural Networks,« 21 7 2017. [Mrežno]. Available: https://users.cecs.anu.edu.au/~sgould/papers/dicta16-depthdropout.pdf.