Umjetne neuronske mreže u prepoznavanju teksta

Fraćin, Mario Branimir

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mario Branimir Fraćin

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Prof. dr. sc. Dubravko Majetić

Student:

Mario Branimir Fraćin

Zagreb, 2019.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija, navedenu literaturu te internet.

Zahvaljujem se prof. dr. sc. Dubravku Majetiću, mentoru ovoga rada, na stručnoj pomoći pri izradi ovog rada.

Mario Branimir Fraćin

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Mario Branimir Fraćin** Mat. br.: 0035205982

Naslov rada na hrvatskom jeziku: **Umjetne neuronske mreže u prepoznavanju teksta**

Naslov rada na engleskom jeziku: **Artificial neural networks in text recognition**

Opis zadatka:

Umjetne neuronske mreže se kao jedan od četiri stupa tehničke umjetne inteligencije uspješno koriste u rješavanju problema prepoznavanja pisanog testa i pisanih brojeva. Prepoznavanje teksta posebno je interesantno područje istraživanja u modernoj kognitivnoj robotici.

U radu treba načiniti sljedeće:

1. Opisati problem prepoznavanja pisanih slova i brojeva.
2. Pokazati stanje u području primjene umjetnih neuronskih mreža u prepoznavanju slova i brojeva.
3. Analizirati i usporediti algoritme umjetnih neuronskih mreža koji se najčešće koriste u tom području istraživanja.
4. Potrebnu programsku podršku pisati u programskom paketu Python i pripadajućoj biblioteci TensorFlow.
5. Izvesti zaključke rada.

Zadatak zadan:
29. studenog 2018.

Rok predaje rada:
1. rok: 22. veljače 2019.
2. rok (izvanredni): 28. lipnja 2019.
3. rok: 20. rujna 2019.

Predviđeni datumi obrane:
1. rok: 25.2. - 1.3. 2019.
2. rok (izvanredni): 2.7. 2019.
3. rok: 23.9. - 27.9. 2019.

Zadatak zadao:
Prof.dr.sc. Dubravko Majetić

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Opis
w	Težina veze između dva neurona
x	Ulazna vrijednost neurona
b	Prag ili pomak neurona (*eng.bias*)
σ	Aktivacijska funkcija
z	Izlazna vrijednost neurona
y	Stvarna vrijednost podatka
$\hat{y}$	Izlazna vrijednost mreže
n	Broj koraka učenja
C	Funkcija cilja (*eng. loss function*)
α	Koeficijent brzine učennja
v	Vrijednost gradijenta prijašnjeg koraka učenja
η	Koeficijent momentuma

SAŽETAK

Tema ovog završnog rada je upotreba umjetnih neuronskih mreža pri prepoznavanju rukom pisanih brojeva i teksta. Uvodno poglavlje daje uvid u sam problem, te ukratko opisuje strukturu i podjelu umjetnih neuronskih mreža. U drugom poglavlju je sam problem prepoznavanja brojki i slova detaljnije opisan, i uz njega su prikazana područja primjena u kojima se danas koriste takvi algoritmi. Zatim je provedena analiza najčešće korištenih algoritama u opisanom području istraživanja. Algoritmi dubokog učenja korišteni u programskoj podršci ovog rada su detaljnije obrađeni u trećem poglavlju, a za ostale suvremene algoritme je dan pregled visoke razine te je ukratko opisan njihov način rada. Konačno, u zadnjem poglavlju su izneseni rezultati rada različitih struktura umjetnih neuronskih mreža za rješavanje navedenog problema koristeći varijacije značajnih parametara. Modeli spomenutih struktura kreirani su u programskom paketu Python koristeći biblioteku TensorFlow uz razne dodatne biblioteke.

Ključne riječi: Umjetna inteligencija, strojno učenje, neuronske mreže, prepoznavanje brojki i teksta, digitalizacija teksta, obrada slike

SUMMARY

The topic of this thesis is the application of artificial neural networks in handwritten character and digit recognition. The problem is presented in the introductory chapter, along with types and structures of artificial neural networks. Second chapter gives a deeper insight to the problem of digit and character recognition, and describes contemporary applications of such technologies. The analysis of most commonly used algorithms in the described field of research is given afterwards. Deep learning methods used in this thesis are thoroughly analyzed in the third chapter, along with a high level overview of other contemporary algorithms and their operating principles. Finally, the last chapter contains the results of different structures of neural networks for the given problem with variations to the most important parameters. Models of previously mentioned structures are created in the programming language Python combined with a machine learning library TensorFlow, along with various additional libraries.

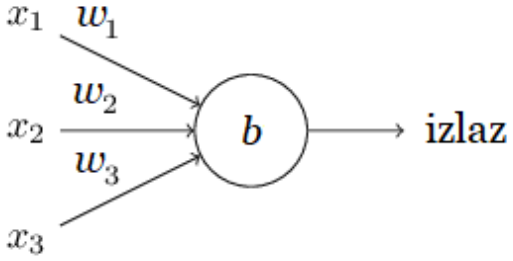
Key words: Artificial intelligence, machine learning, neural networks, digit and text recognition, text digitalisation, image processing

1. UVOD

Prepoznavanje rukom pisanog teksta i brojeva je za čovjeka toliko trivijalan zadatak da ga svakodnevno rješava bez posebnog napora. Iako su milijuni godina evolucije omogućile ljudskim bićima da takav problem rješavaju gotovo instantno, programiranje računala da učini isto, nije tako jednostavno. Navedeni problem je beznadno pokušati riješiti klasičnim putem, pisanjem programskog koda s pravilima i iznimkama, zbog previše različitih informacija koje mogu biti sadržane u jednoj slici. Upravo zato su se počele koristiti umjetne neuronske mreže koje funkcioniraju na način da se umjesto posebnog unošenja pravila, računalu da dovoljno primjera s pripadajućim rješenjima te se pusti da stvori vlastiti set pravila. Sam postupak podrazumijeva prikupljanje slika, obradu, slaganje i označavanje istih, treniranje mreže te konačno prepoznavanje slova i brojeva. Navedeni problem spada u područje istraživanja procesiranja slika i uzoraka, u kojem je cilj „naučiti“ računalo da dostigne gotovo ljudsku razinu prepoznavanja teksta, pa se time otvaraju mogućnosti za automatizaciju različitih postupaka poput bankarskih transakcija, slaganja pošte, kategoriziranje dokumenata i slično.

1.1. Umjetni neuron

Osnovna gradivna jedinica umjetnih neuronskih mreža je naravno umjetni neuron. Poput mnogih modernih izuma i proizvoda, inspiracija za umjetni neuron pronađena je u prirodi te je modeliran prema biološkom neuronu. Sredinom 20. stoljeća neuropsiholog Warren McCulloch i matematičar Walter Pitts su opisali način rada biološkog neurona, te su svoje istraživanje potkrijepili modelom jednostavne neuronske mreže koristeći električne komponente [1]. Razvojem računala su se pojavile simulirane jednoslojne neuronske mreže, te ubrzo nakon višeslojne neuronske mreže koje se koriste i danas. Značajniji napredak u razvoju neuronskih mreža bilo je otkriće perceptrona, to je jednostavan oblik neurona prikazan na slici 1.1.



Slika 1.1. Perceptron

Perceptron funkcionira tako da preko ulaza x_j prima binarne vrijednosti, množi ih s vrijednostima težina w_j te ovisno o vrijednosti praga b (*eng. bias*) daje izlaz 0 ili 1. Matematički zapis toga glasi:

$$\text{izlaz} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \tag{1.1}$$

gdje su:

$$w \cdot x = \sum_j w_j \cdot x_j \tag{1.2}$$

Odnosno veći prag b rezultira većom vjerojatnošću aktivacije neurona i obrnuto. Poznavanje rada jednostavnog perceptrona olakšava shvaćanje rada neurona u suvremenoj primjeni.

Suvremene neuronske mreže koriste neurone sa aktivacijskim funkcijama od kojih je najpoznatiji neuron sa sigmoidnom aktivacijskom funkcijom. Takav neuron također ima težine w_j, prag b, i ulaze x_j, ali za razliku od perceptrona ulazne vrijednosti su brojevi između 0 i 1. Dodatna razlika je da izlaz sigmoidnog nije striktno binaran, već je definiran aktivacijskom funkcijom:

$$\sigma(w \cdot x + b) \tag{1.3}$$

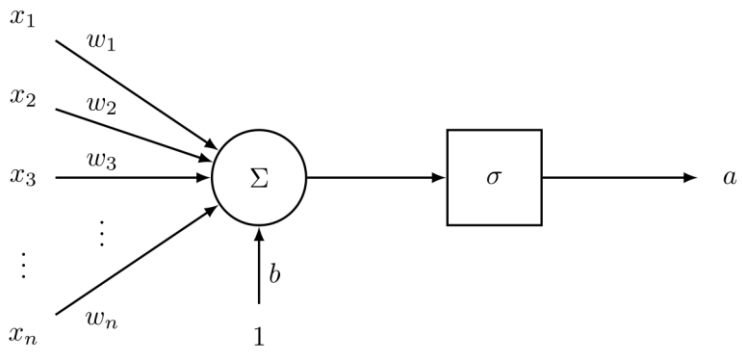
Gdje je σ jednak:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.4}$$

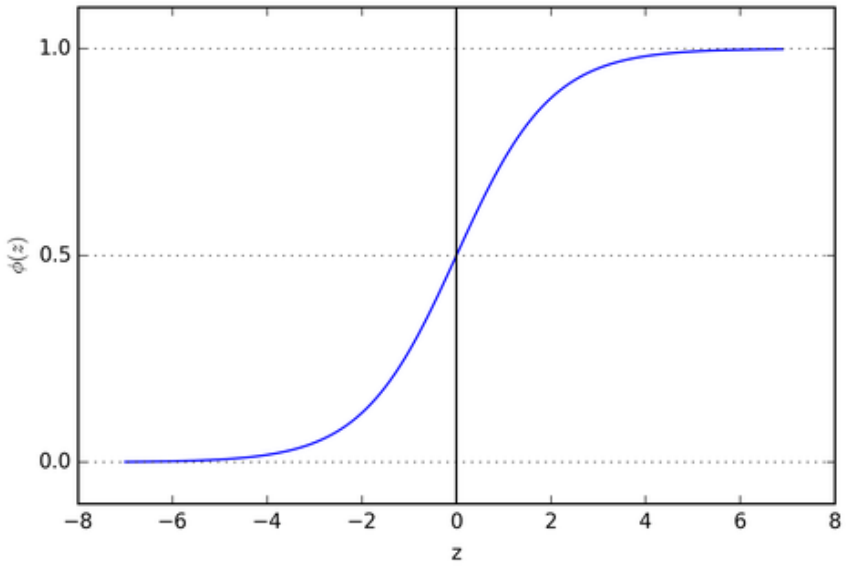
A z:

$$z = (w \cdot x + b) \tag{1.5}$$

Iz gore navedenih jednadžbi je vidljivo da su izlazi neurona vrijednosti između 0 i 1. Sigmoidni neuron prikazan je na slici 1.2, a njegova aktivacijska funkcija na slici 1.3.



Slika 1.2. Sigmoidni neuron



Slika 1.3. Sigmoidna funkcija

Prednost sigmoidne aktivacijske funkcije je ta što mala promjena ulaza *z* rezultira malom promjenom izlaza *σ(z)*, a upravo takva karakteristika pomaže u efikasnijem radu neuronske mreže. Uz sigmoidnu funkciju danas se koriste i mnoge druge zavisno o primjeni mreže. Primjeri takvih funkcija i njihovi matematički zapisi prikazani su u tablici 1.1.

Tablica 2.1. Tipovi aktivacijskih funkcija

Aktivacijska funkcija	Jednadžba
Linearna funkcija	$f(x) = x$
Sigmoidna funkcija	$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$
Funkcija tangensa hiperbolicnog	$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
ReLu	$f(x) = \begin{cases} 0 \text{ za } x < 0 \\ x \text{ za } x \geq 0 \end{cases}$
Softmax funkcija	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

U programskoj podršci je korištena ReLu funkcija, te je ista detaljnije obrađena u nastavku rada.

U ovom podnaslovu je ukratko opisan umjetni neuron i njegov način rada, dok će u sljedećim poglavljima isti biti detaljnije obrađen kao i umjetne neuronske mreže koje neuroni čine.

1.2. Podjela umjetnih neuronskih mreža

Umjetne neuronske mreže se dijele po različitim kriterijima od kojih je najjednostavniji broj slojeva. Mreže se sastoje od ulaznog sloja, izlaznog sloja te skrivenih slojeva. S obzirom na broj skrivenih slojeva dijele se u:

- jednoslojne,
- dvoslojne,
- troslojne
- višeslojne.

Sljedeći kriterij po kojem se dijele umjetne neuronske mreže je tok signala, pa tako mogu biti unaprijedne (*eng. Feedforward*) ili povratne (*eng. Recurrent*). Kod unaprijednih mreža kretanje signala je isključivo u jednom smjeru iz prethodnog sloja u sljedeći, dok se kod povratnih mreža signal vraća na početak te se uzima u obzir pri učenju mreže.

Umjetne neuronske mreže se također mogu podijeliti prema načinu učenja:

- mreže s učiteljem (*eng. Supervised learning*). Takve mreže posjeduju podatke iz kojih mreža uči i njihova komplementarna rješenja (*eng. Labels*), pa se tako predviđanja mreže mogu usporediti sa stvarnim vrijednostima nakon faze učenja.
- mreže bez učitelja (*eng. Unsupervised learning*), ne posjeduju podatke te same stvaraju kategorije značajki, pa se takve mreže koriste pri grupiranju podataka.
- mreže s pojačanim učenjem (*eng. Reinforcement learning*), funkcioniraju na temelju metode pokušaja i pogreške. Mreža isprobava radnje koje daju najveću nagradu prema nekakvoj metrici, i prema tome se prilagođava.

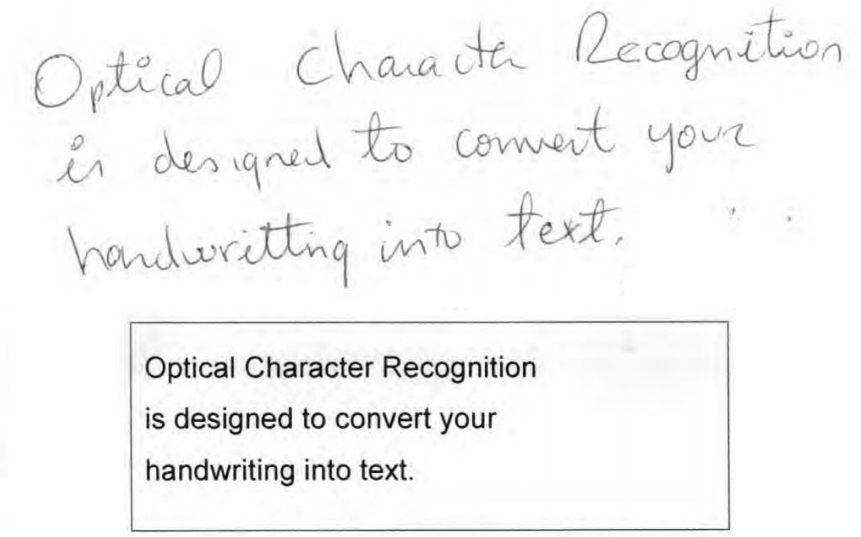
Također postoji podjela prema broju koraka, pa postoje mreže koje uće iterativno i one koje uče u jednom koraku.

Konačna podjela je prema pravilu učenja, iako danas postoji mnogo pravila ovdje će biti navedene samo najčešće korištene, a to su:

- konvolucijske mreže
- Adaline mreže
- Madaline mreže
- Hopfieldove mreže
- Boltzmanove mreže

2. PRIMJENA UMJETNIH NEURONSKIH MREŽA U PREPOZNAVANJU SLOVA I BROJEVA

Svaka moderna industrija teži povećanjem efikasnosti, a najčešće to postiže automatizacijom rada. Uvođenjem automatskih sustava smanjuje se broj potrebnih radnika, skraćuje se vrijeme proizvodnje odnosno stvaranja usluge, povećava se sigurnost pojedinaca na radnom mjestu i u konačnici stvara ušteda u konačnom proizvodu ili usluzi. Među sredstvima automatizacije rada u raznim birokratskim i industrijskim okruženjima primjenu su našli sustavi prepoznavanja slova i brojeva temeljeni na umjetnoj inteligenciji. Slika 2.1. prikazuje primjer rada jedne takve mreže. Skeniranjem rukom pisanog teksta koristeći kameru ili skener, dokumenti se pretvaraju u digitalni oblik te se uz minimalnu predobradu mogu koristiti sa algoritmima umjetne inteligencije.



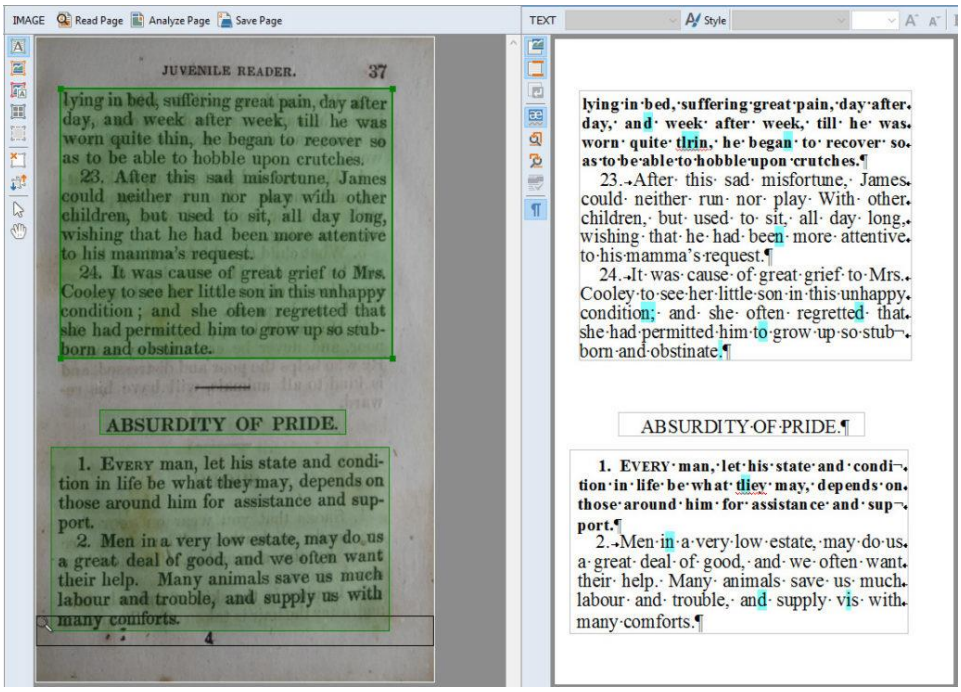
Slika 2.1. Digitalizacija rukom pisanog teksta

2.1. Područja primjene

Logično je da će se navedena tehnologija koristiti u poslovima koji obrađuju velike količine pisanih dokumenata, a dvije najveće takve djelatnosti su bankarstvo i sudstvo[2]. Skladištenje velike količine fizičkih dokumenata znatno otežava pretragu informacija, smanjuje iskoristivost prostora i povećava mogućnost gubitka informacija. Bankarski sektor pretežito koristi navedenu tehnologiju za provjeravanje čekova, osobnih isprava i potpisa. Time se ubrzavaju procesi registracija, potpisivanja dokumenata i transakcija. Iako proces nije u potpunosti automatiziran, stvara primjetnu uštedu vremena od kojih beneficira poduzeće, a time i krajnji potrošač.

Sudstvo je jedno od područja koje bi moglo imati znatnu korist od navedene tehnologije. Različiti dokumenti poput presuda, zahtjeva, oporuka i izjava, mogu biti digitalizirani i time znatno dostupniji. Kao i kod bankarskog sustava, stvara se primjetna ušteda vremena i veća transparentnost podataka.

Osim obrade službenih dokumenata navedeni sustavi mogu digitalizirati tekstove pisane različitim pismom, poput Kineskog pisma, Arapskog pisma i Ćirilice [3]. Na taj način se osigurava dugovječnost pisanih dijela, a i lakše prevođenje istih, što rezultira povezivanjem svijeta i kultura. Primjer skeniranja ispisanog dijela prikazan je na slici 2.2.



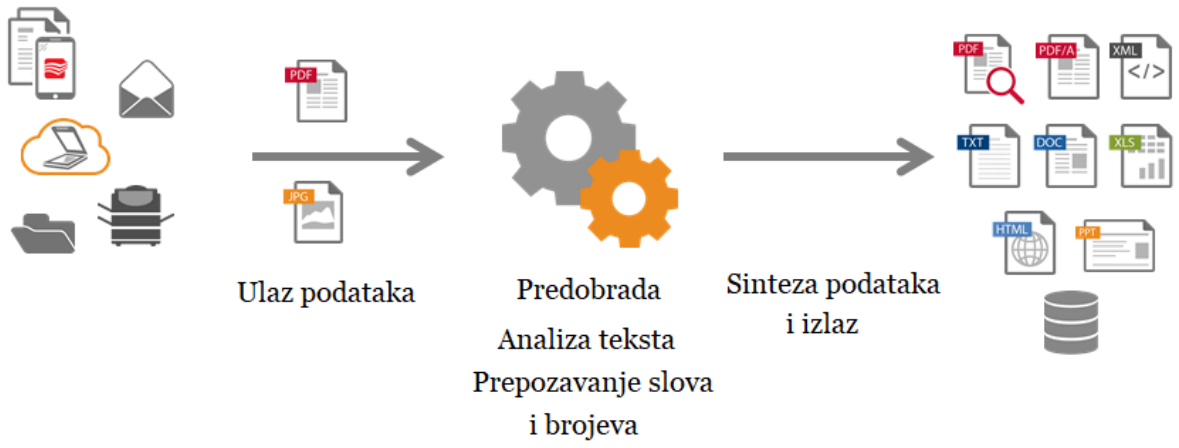
Slika 2.2. Digitalizacija ispisanog dijela

Uz navedeno, umjetne neuronske mreže su primjenu našle u zdravstvenoj industriji, gdje se zbog pojednostavljenje pregleda medicinskih dokumenata lakše postavljaju dijagnoze, čime se potencijalno mogu spasiti ljudski životi.

Osim spomenutih, naveden suvremene primjene uključuju prevođenje prometnih znakova u stvarnom vremenu, pomoć pri autonomnoj vožnji, razvrstavanje proizvoda ili dokumenata te poboljšanu kvalitetu životu slabovidnim i slijepim osobama. Jasno je da primjena ove tehnologije, osim što povećava stupanj automatizacije određenih procesa, nudi rješenja svakodnevnim problemima za koje se nekad smatralo da ih samo čovjek može riješiti.

2.2. Prednosti umjetnih neuronskih mreža pri digitalizaciji teksta

Ovdje će biti prezentiran sažetak beneficija koje pružaju tehnologije spomenute u prethodnom podnaslovu, neovisno o području primjene. Primarne prednosti takvih sustava su mogućnost pretrage, uređivanja i pristupa. Jednostavnom pretvorbom u univerzalne formate poput PDF-a, ključni pojmovi mogu instantno biti pronađeni unutar teksta. U slučaju da originalni dokument treba urediti, umjesto prepisivanja čitavog teksta iznova, skenirana datoteka se jednostavno otvori u programskom paketu za obradu teksta i uredi se željeni dio. Učitavanjem dokumenata na zajednički server ili „oblak“, institucije i tvrtke imaju trenutni pristup podacima neovisno o njihovoj fizičkoj lokaciji. To rezultira većom iskoristivošću fizičkog prostora i smanjenjem upotrebe papira, što daje i „zelenu“ notu spomenutoj tehnologiji. Također, instantna dostupnost podataka omogućava provođenje analize i obrade podataka, i na taj način osigurava točnost podataka i predviđanje trendova poslovanja. Za razliku od fizičkih dokumenata moguće je stvoriti kopije digitalnih dokumenata na različitim virtualnim lokacijama i na taj način osigurati sigurnost podataka i njihovo vraćanje u slučaju nezgode. Kao što je spomenuto u prijašnjem podnaslovu, ako dokumenti već postoje u digitalnom obliku vrlo se lako mogu upotrijebiti u programima za prevođenje teksta, time uklanjajući kulturološke i poslovne barijere. Slika 2.3. prikazuje pojednostavljeni postupak pretvaranja pisanog dokumenta u digitalni. Postupak podrazumijeva nabavu podataka, koja može biti obavljena skeniranjem, slikanjem ili slanjem podataka sa drugih lokacija. Zatim se podaci pretvaraju u prikladne formate za obradu, nakon čega slijedi obrada, analiza teksta te prepoznavanje znakova. Konačan rezultat su izlazne datoteke željenog formata.



Slika 2.3. Postupak digitalizacije teksta

3. SUVREMENI ALGORITMI UMJETNE INTELIGENCIJE KORIŠTENI PRI PREPOZNAVANJU PISANIH SLOVA I BROJEVA

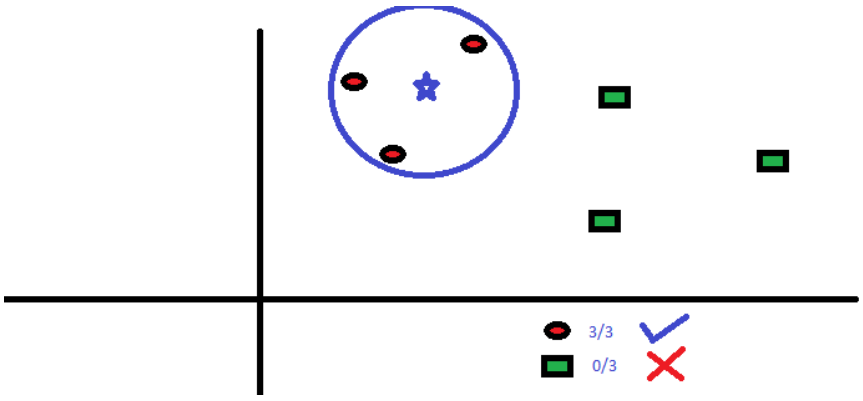
Za rješavanje navedenog problema danas se koriste klasifikatorski algoritmi poput [4]:

- algoritma K – najbližeg susjeda
- metoda potpornih vektora
- neuronske mreže
- konvolucijske neuronske mreže

U ovom poglavlju će biti dan teoretski pregled prve dvije metode, dok će krajnje dvije metode dubokog učenja biti detaljnije analizirane. Razlog tomu je taj da algoritam K – najbližeg susjeda i metoda potpornih vektora nisu korišteni u programskoj podršci ovog rada.

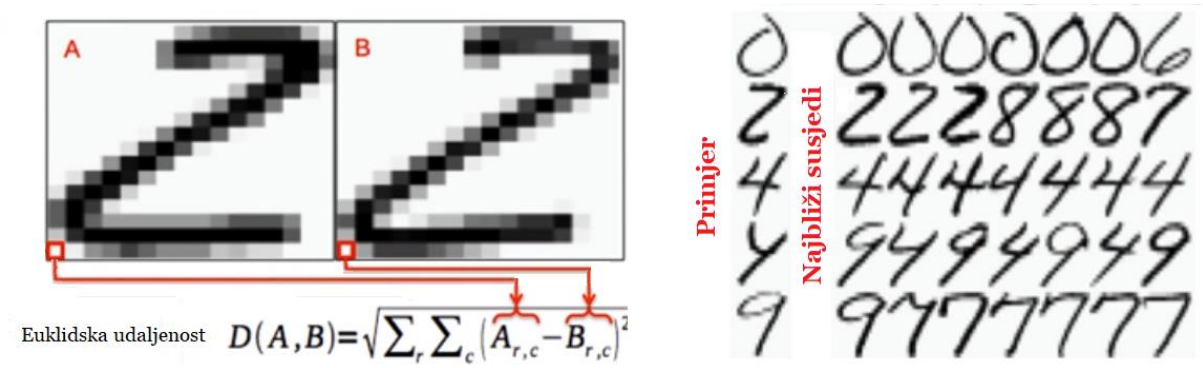
3.1. Algoritam K – najbližeg susjeda

Algoritam K – najbližeg susjeda poznat je i pod imenom kNN. S obzirom na jednostavnost rada, točnost je iznenađujuće visoka i iznosi 95% pri prepoznavanju rukom pisanih brojki. Algoritam funkcionira na principu prepoznavanja sličnosti značajki, odnosno podaci za treniranje koriste se u fazi testiranja. Najbitniji parameter ove metode je vrijednost *k,* kojom se određuje količina podataka prema kojima će se raditi klasifikacija. Primjer prikazan na slici 3.1 simbolički prikazuje klasifikaciju podatka kNN algoritma. Na temelju udaljenosti od ostalih najbližih podataka određuje da li novi podatak pripada skupini koje obilježavaju crveni krugovi ili zeleni kvadrati. Ako se uzme da je *k* jednak 3, pomoću funkcija udaljenosti se određuju 3 najbliža podatka koja su u ovom slučaju crveni krugovi, te algoritam klasificira plavu zvijezdu kao crveni krug.



Slika 3.1. Prikaz jednostavnog kNN algoritma

Na identičan način funkcionira prepoznavanje slova i brojki, jedina razlika je ta da se umjesto geometrijskih oblika uspoređuju pojedini pikseli slika. Uspoređujući vrijednosti sivih tonova pojedinih piksela korištenjem euklidske funkcije udaljenosti, algoritam predviđa vrijednost brojke. Vrijednost *k* određuje broj predviđanja nakon kojih se donosi konačna odluka. Gore navedeni postupak prikazan je slikom 3.2., na kojoj se s lijeve strane nalazi postupak usporedbe, a s desne postupak odlučivanja.



Slika 3.2. kNN algoritam pri prepoznavanju brojki

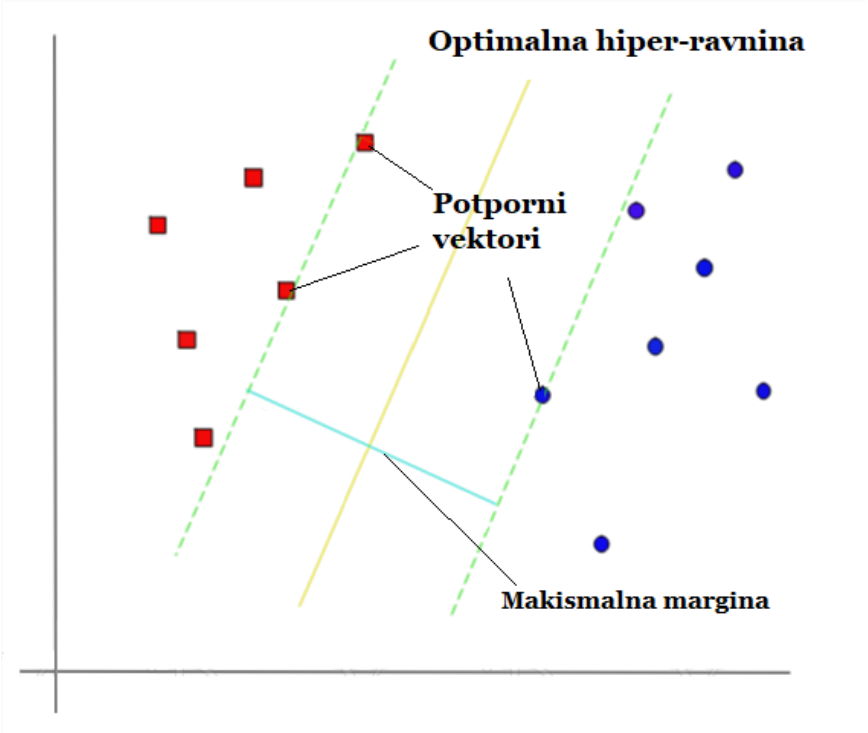
Prednost ovog algoritma je jednostavnost implementacije i fleksibilnost pri rješavanju problema regresije i klasifikacije. Uz to nije potrebno izraditi model i podešavati veliki broj parametara kao što je slučaj kod dubokih neuronskih mreža. Jednostavnost algoritma ograničava točnost istog, jer povećanjem raznolikosti primjera i broja varijabli točnost drastično opada.

3.2. Metoda potpornih vektora

Uz rastući trend korištenja umjetnih neuronskih mreža za rješavanje različitih kompleksnih problema, metoda potpornih vektora je često previđena, osobito u području obrade slike. Zahvaljujući brojnim programskim bibliotekama poput *scikit.learn* biblioteke, algoritam je moguće implementirati u svega nekoliko linija koda te koristeći različita poboljšana, postići nevjerojatnu točnost od 98% na zadatku prepoznavanja rukom pisanih brojki. U nastavku će ukratko biti opisan način rada takvog algoritma, no detaljnije matematičko objašnjenje će biti izostavljeno jer nadilazi opseg ovog rada. Slika 3.3. prikazuje osnovne elemente poput hiper-ravnine, potpornih vektora i maksimalne margine. Na osima grafa se naravno nalaze značajke

po kojima se vrši podjela podataka. Suština rada algoritma je ta da se pokušava naći hiper-ravninu, odnosno funkciju koja će efikasno podijeliti dvije ili više skupine podataka.

U slučaju prikazanom na slici 3.3 ta funkcija je pravac, no u praksi može biti u obliku polinoma višeg reda ili trigonometrijske funkcije. Kako se funkcija ne bi postavljala proizvoljno, koriste se podaci na rubovima pojedinih skupina pomoću kojih se određuje maksimalna margina između dvije skupine. Takvi podaci se nazivaju potporni vektori i upravo po tome je algoritam dobio ime.



Slika 3.3. Vizualni prikaz metode potpornih vektora

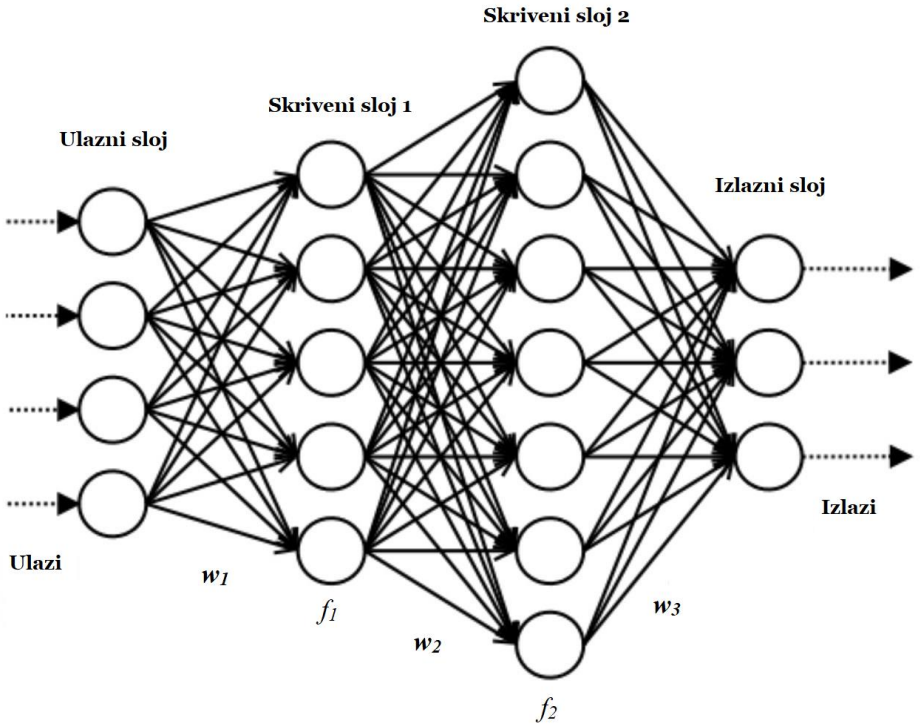
U slučaju prepoznavanja brojki i slova, pojedine skupine podataka označavaju odgovarajuće brojeve i slova. U fazi treniranja se postavljaju hiper-ravnine koje odjeljuju navedene skupine, dok se u fazi testiranja novi podatak klasificira na temelju lokacije u odnosu na postavljene hiper-ravnine. Metodu potpornih vekora odlikuje visoka točnost uz efikasnu potrošnju digitalnih resursa te visoka efektivnost pri obradi više-dimenzijskih podataka. Mane algoritma su te da kod velikih skupova podataka treniranje zahtjeva mnogo vremena, te kod podataka s puno „šuma“ i preklapanja točnost znatno opada.

3.3. Umjetne neuronske mreže

U prijašnjim podnaslovima su obrađeni jednostavniji algoritmi umjetne inteligencije, koji usprkos svojoj jednostavnosti daju zadovoljavajuće rezultate. Ovaj i sljedeći podnaslov će dati uvid i analizu u metode dubokog učenja koje su ustvari sama srž ovog rada. Za usporedbu, neuronske mreže mogu dati točnosti u prepoznavanju brojki do 99%, no velika točnost se dobiva na račun velike kompleksnosti mreže, a time i uporabom velike količine računalnih resursa i velikih vremena učenja. U uvodnom poglavlju je dan uvid u pojednostavljeni način rada neuronskih mreža, dok će ovdje ista tema biti detaljnije obrađena. U sljedećim podpodnaslovima će biti dan opis strukture mreže i sam način na koji mreža „uči“.

3.3.1. Struktura umjetne neuronske mreže

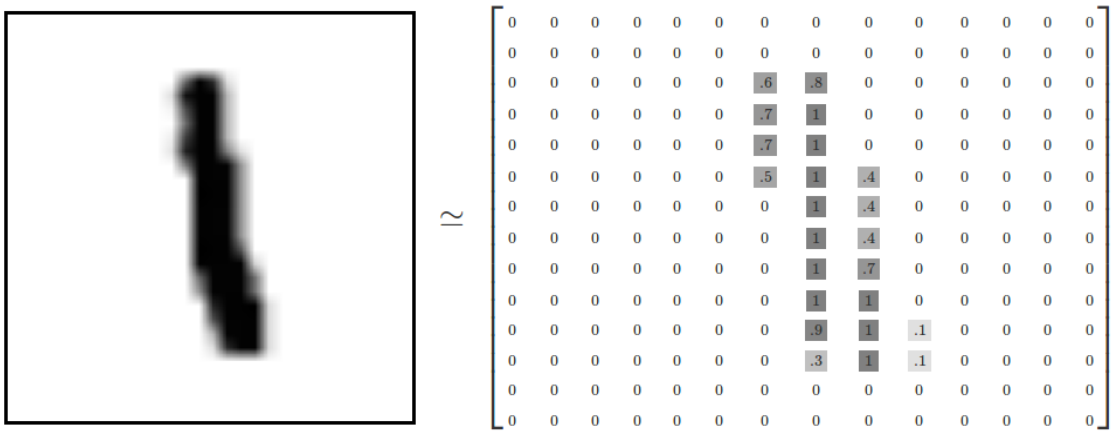
Neuronska mreža se sastoji od međusobno povezanih neurona, koji su u suštini matematičke funkcije gdje su izlazne vrijednosti funkcija jednog sloja jednake ulaznim vrijednostima funkcija drugog sloja. Slika 3.4. prikazuje strukturu jednostavne neuronske mreže s 4 ulaza, 2 skrivena sloja te 3 izlaza. Između svakog od slojeva se nalazi vrijednost $\boldsymbol{w_n}$, koja označava vektor vrijednosti težina pojedinih veza. Oznake f_1 i f_2 predstavljaju aktivacijske funkcije neurona u pripadajućim slojevima.



Slika 3.4. Struktura neuronske mreže

Svaki „krug“ predstavlja neuron s određenom aktivacijskom funkcijom prikazanom u tablici 1.

U slučaju prepoznavanja brojki i slova, broj ulaznih neurona mreže je jednak broju piksela sadržanih u slici brojke ili slova. Slika 3.5. prikazuje matrični prikaz rukom pisanog broja 1 rezolucije 14x14 piksela. Svaki piksel odnosno neuron sadržava vrijednost u rasponu između 0 i 1 ovisno o svjetlini. Vrijednost 0 označava potpuno osvijetljen piksel, odnosno bijelu boju, dok vrijednost 1 označava potpuno zatamnjen piksel, odnosno crnu boju.



Slika 3.5. Matrični prikaz rukom pisane znamenke

Prema navedenom neuronska mreža koja prepoznaje rukom pisane znamenke sa slika dimenzija 14x14 piksela bi imala ulazni sloj sa 196 neurona, a izlazni sloj bi se sastojao od 10 neurona koji bi „donosio“ konačnu odluku o kojoj se znamenci radi. Za jednostavne probleme je često dovoljan samo 1 skriveni sloj, dok se za većinu problema koriste 2 skrivena sloja. Broj skrivenih slojeva i neurona u svakom od njih se uglavnom odabire proizvoljno te se povećanjem broja istih točnost mreže značajno poboljšava. Kao primjer se može navesti mreža trenirana za prepoznavanje rukom pisanih brojeva s greškom od 0.35%. Takva mreža se sastoji od 6 skrivenih slojeva, gdje broj skrivenih neurona po sloju može dostići i 2500 [5]. Iako veliki broj neurona u skrivenom sloju postiže veću točnost mreže, također može utjecati negativno. Do toga dolazi kada mreža savršeno nauči primjere iz seta podataka za treniranje, pa daje loše performanse kada joj se predstave stvarni primjeri, taj se problem se naziva prekomjerna specijalizacija modela (*eng. overfitting*). Pošto mreže slične navedenoj sadržavaju veliki broj neurona, težina i pragova, jednadžba 1.4 prelazi u matrični oblik koji glasi:

$$\sigma\left(\begin{bmatrix} w_{0,0} & \cdots & w_{0,n} \\ \vdots & \ddots & \vdots \\ w_{k,0} & \cdots & w_{k,n} \end{bmatrix}\begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}\right) \tag{3.1.}$$

3.3.2. Princip učenja umjetne neuronske mreže

Pošto za problem prepoznavanja pisanog teksta i brojki točnost neuronskih mreža može biti vrlo visoka, za očekivati je da su nadležne matematičke operacije vrlo komplicirane. U stvarnosti, čitav princip funkcioniranja umjetnih neuronskih mreža se temelji na osnovama vektorskog, matričnog, te diferencijalnog računa. Za primjer prikazan slikom 3.5. uz dva skrivena sloja po 10 neurona, mreža je u suštini funkcija s 2190 parametara. Do toga broj se dolazi zbrajajući sve težine uključujući one koje povezuju Bias neurone, te je očito da taj broj znatno raste povećanjem broja neurona i skrivenih slojeva. Proces učenja umjetne neuronske mreže se sastoji od podešavanja navedenih parametara tako da mreža daje zadovoljavajuće rezultate. Naravno potrebno je uvesti nekakvu metriku na temelju koje bi se procjenjivala točnost mreže, a u ovom slučaju ona dolazi u obliku funkcije cilja (*eng. loss function*).

Funkcija cilja definira razliku željene vrijednost i dobivene vrijednosti izlaznog sloja neuronske mreže, te u velikom broju slučaja učenje same mreže se temelji na minimiziranju iste. Drugim riječima, izlaz funkcije cilja je vrijednost koja označava koliko je mreža precizna u obavljanju određenog zadatka za zadane parametre, te njen minimum označava optimalne performanse mreže. Ne postoji funkcija cilja koja je savršena za svaki slučaj učenja, tako da ovisno o zadatku treba odabrati funkciju koja je prikladna problemu koji se rješava. Pri rješavanju problema regresije, jedna od najčešćih funkcija je funkcija srednje kvadratne pogreške (*eng. Mean-Squared* error) čija jednadžba glasi:

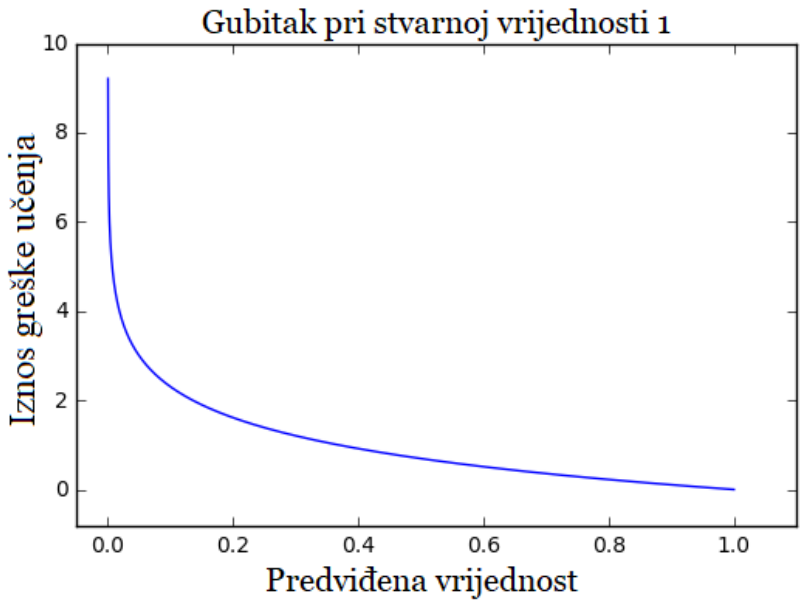
$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{3.2.}$$

U navedenoj jednadžbi *n* označava broj podataka za testiranje, *i* indeks trenutnog podatka za testiranje, y_i stvarnu vrijednost podatka na i-tom mjestu, te konačno $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ predviđenu vrijednost mreže. Karakteristika navedene funkcije je ta da veća razlika između ulaznih vrijednosti rezultira znatnije većim iznosom greške. Osim spomenute, za rješavanje problema regresije još se koriste funkcije srednje apsolutne vrijednosti (*eng. Mean Absolute Error*) te srednje vrijednosti praga (*eng. Mean Bias Error*).

Najčešća funkcija cilja kod problema klasifikacije je funkcija ukrštene entropije (eng. Cross entropy) i glasi:

$$C = -\frac{1}{n}\sum_{i=1}^{n}[y_i \, ln \, \hat{y}_i + (1 - y_i) \, ln(1 - \hat{y}_i)] \tag{3.3.}$$

Slika 3.6. prikazuje način na koji navedena funkcija daje vrijednost greške kod mreže za problem prepoznavanja brojki. Iznos greške učenja se mijenja po logaritamskoj funkciji, pa se tako rezultatima daleko od stvarne vrijednosti dodjeljuje visoki iznosi greške, a onim predviđanjima blizu stvarne vrijednosti iznosi greške su gotovo nula.



Slika 3.6. Funkciju ukrštene entropije

Uz navedenu funkciju pri rješavanju problema klasifikacije također može poslužiti „Hinge-Loss“ funkcija. Ona funkcionira na način da mreža daje određene vrijednosti pri prepoznavanju različitih slika, te bi zbroj vrijednosti točne kategorije trebao biti veći od zbroja vrijednosti netočnih kategorija.

Postupak učenja mreže se svodi na minimiziranje neke od gore spomenutih funkcija cilja, pa je tako cilj učenja smanjiti razliku između predviđene i stvarne vrijednosti. Brojka 1 prikazana slikom 3.5 sadržava preveliki broj varijabli da bi se pripadajuća funkcija cilja mogla vizualizirati, iz tog razloga će primjer učenja mreže biti prikazan funkcijom s jednom varijablom. Takva funkcija *C(w)* je prikazana na slici 3.7., te je nasumično odabrana početna težina prikazana crvenom točkom. Pošto je u cilju težiti minimumu funkcije cilja, logičan pristup je naći derivaciju odnosno gradijent, te za određeni korak pomaknuti težinu *w* u smjeru suprotno derivacije. Takav algoritam pomicanja u smjeru suprotno od gradijenta se naziva gradijentni spust (*eng. Gradient descent*) i najčešći je algoritam smanjivanja greške.

Jednadžba algoritma glasi:

$$w_{j+1} = w_j - \alpha \nabla C(w_j) \tag{3.4.}$$

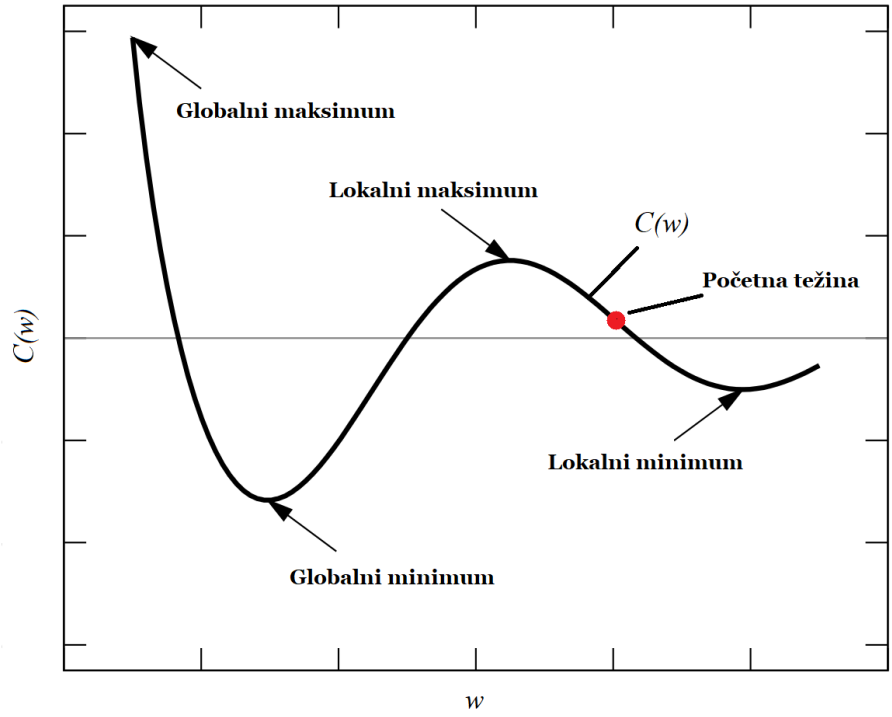
Gdje indeks j označava poziciju težine w u vektoru $\boldsymbol{w}$, α koeficijent brzine učenja i $\nabla C(w_j)$ gradijent odnosno najveću promjenu funkcije po varijabli w. Prema tome gradijent funkcije cilja jednak je:

$$\nabla C(w_j) = \frac{\partial C(w)}{\partial w_j} \tag{3.5.}$$

Navedeni algoritam se ponavlja sve dok se težina ne zaustavi u jednom od minimuma odnosno dok gradijent funkcije cilja ne postane jednak 0:

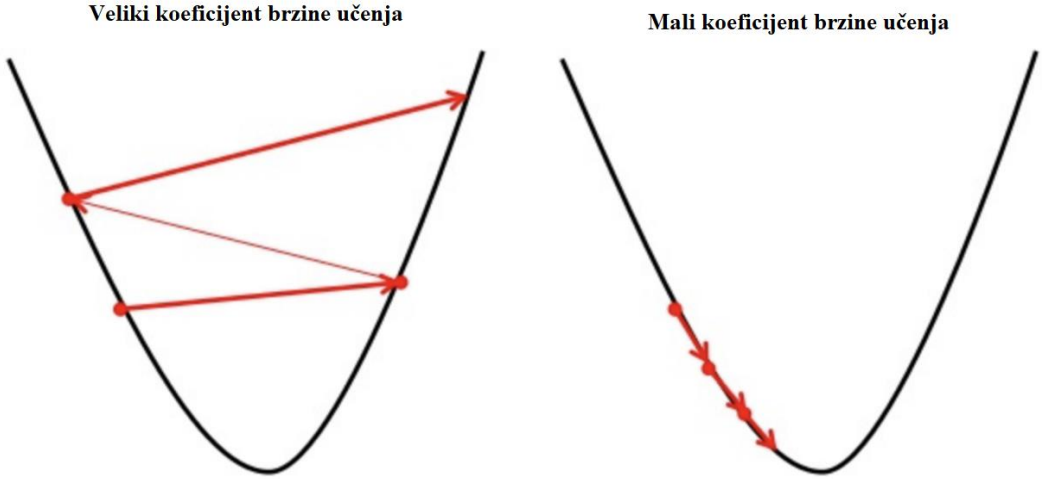
$$\nabla C(w) = 0 \tag{3.6.}$$

Ovakav jednostavan zadatak daje uvid u kompleksnost problema nalaženja globalnog minimuma čak i za funkcije s jednom varijablom, s obzirom da je vrlo lako moguće zapeti u lokanom minimumu. Problem postaje znatno kompliciraniji uvođenjem više varijabli i time povećavajući broj dimenzija.



Slika 3.7. Vizualni prikaz funkcije cilja sa jednom težinom

U jednadžbi (3.4). je uveden faktor α koji označava koeficijent brzine učenja, te je jedan od glavnih parametara pomoću kojeg se može utjecati na konvergenciju rješenja mreže. Najčešće vrijednosti faktora nalaze se u rasponu između 0 i 1, te se zavisno o odabiru koeficijenta α može dogoditi jedna od 3 moguće 3 situacije. Pri odabiru malog iznosa koeficijenta učenja, smanjuje se mogućnost premašivanja minimuma, ali rješenje sporo konvergira pa je za točnost mreže potrebno osigurati dovoljan broj koraka učenja. Kod odabira velikog koeficijenta brzine učenja može se doći do alternirajućeg približavanja rješenju ili u najnepovoljnijem slučaju divergenciji rješenja. Navedeni slučajevi prikazani su na slici 3.8.



Slika 3.8. Vizualni prikaz malog i velikog koeficijenta brzine učenja

Kao nadogradnja na spomenuti algoritam, često se uvodi koeficijent momentuma ili zamaha, koji uz koeficijent brzine učenja, osigurava da algoritam ne „zaglavi“ u lokalnom minimumu te može ubrzati nalaženje minimuma pri malim koracima. U programskoj podršci tako modificirani algoritam se naziva momentni optimizator (*eng. Momentum optimizer*), te njegova jednadžba glasi [6]:

$$\begin{cases} v_j = \eta v_j - \alpha \nabla C(w_j) \\ w_j = v_j - w_j \end{cases} \tag{3.7.}$$

Paramater η označava koeficijent momentuma te se za njega odabiru vrijednosti između 0 i 1 kao i za koeficijent brzine učenja α. Varijabla v_j sadrži vrijednosti gradijenata korištenih u prijašnjim koracima učenja, te upravo te dvije spomenute varijable čine razliku između klasičnog gradijentnog spusta i ovog algoritma. Veća točnost algoritma uzrokovana je time da se pri optimizaciji u obzir uzimaju i prijašnje vrijednosti gradijenta funkcije cilja.

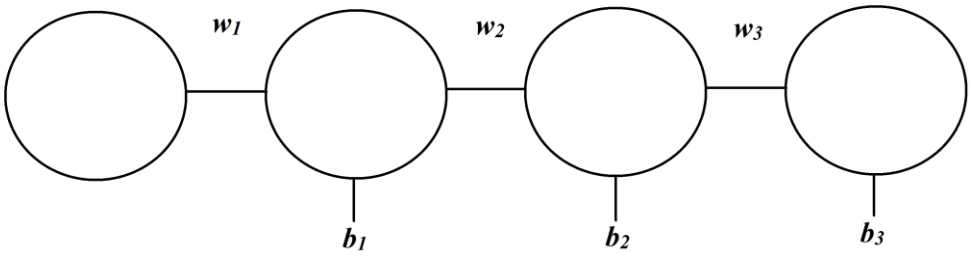
3.3.3. Algoritam povratnog prostiranja pogreške

Ovdje obrađeni algoritam povratnog prostiranja pogreške opisuje finalnu fazu učenja umjetne neuronske mreže. Gore spomenuta metoda gradijentnog spusta se temelji na nalaženju gradijenta funkcije cilja, do koje se dolazi upravo ovim algoritmom. Postupak se sastoji od određivanja parcijalnih derivacija funkcije cilja u odnosu na težine običnih i Bias neurona, pa se navedene derivacije koriste za promjenu istih. Radi jednostavnijeg objašnjenja, za primjer se uzima jednostavna mreža s 4 sloja i po jednim neuronom u svakom od slojeva, prikazana na slici 3.9. Radi dodatnog pojednostavljena fokus će biti samo na zadnja dva sloja odnosno neurona, gdje će izlaz posljednjeg sloja biti označen sa $x^{(s)}$, a pretposljednjeg sloja sa $x^{(s-1)}$ pa jednadžba koje povezuje te dvije vrijednosti glasi:

$$x^{(s)} = \sigma\left(z^{(s)}\right) \tag{3.8.}$$

Gdje je σ aktivacijska funkcija neurona, a $z^{(s)}$:

$$z^{(s)} = w^{(s)}x^{(s-1)} + b^{(s)} \tag{3.9.}$$



Slika 3.9. Jednostavna umjetna neuronska mreža sa 4 sloja i 4 neurona

Vrijednost $x^{(s)}$ je od velikog značaja, jer ona u konačnici ulazi u funkciju cilja te određuje točnost mreže. Iz jednadžbe (3.9.) je očito da je ta vrijednost ovisna o paremetrima težine $w^{(s)}$, aktivaciji neurona prethodnog sloja $x^{(s-1)}$, te praga $b^{(s)}$. Sukladno tome potrebno je naći derivacije funkcije cilja u odnosu na te varijable, te su one komponente vektora gradijenta funkcije cilja $\boldsymbol{\nabla C}$. Do navedenog se dolazi koristeći pravilo lančanog deriviranja, i to će biti prikazano na primjeru derivacije po težini $w^{(s)}$.

$$\frac{\partial C_0}{\partial w^{(s)}} = \frac{\partial z^{(s)}}{\partial w^{(s)}}\frac{\partial x^{(s)}}{\partial z^{(s)}}\frac{\partial C_0}{\partial x^{(s)}} \tag{3.10.}$$

Uzimajući prosjek vrijednosti po svim testnim vrijednostima dobiva se komponenta vektora gradijenta funkcije cilja $\boldsymbol{\nabla C}$ za težine zadnjeg sloja te ona glasi:

$$\frac{\partial \boldsymbol{C}}{\partial w^{(s)}} = \frac{1}{n}\sum_{k}^{n-1}\frac{\partial C_k}{\partial w^{(s)}} \tag{3.11.}$$

Da bi vektor $\boldsymbol{\nabla C}$ bio potpun, isto je potrebno napraviti za sve težine između svih slojeva, težine Bias neurona i izlaze svih neurona pripadajućeg sloja. Analogno jednadžbi (3.10.) lančana derivacija težina Bias neurona glasi:

$$\frac{\partial C_0}{\partial b^{(s)}} = \frac{\partial z^{(s)}}{\partial b^{(s)}}\frac{\partial x^{(s)}}{\partial z^{(s)}}\frac{\partial C_0}{\partial x^{(s)}} \tag{3.12.}$$

Te za izlaze neurona:

$$\frac{\partial C_0}{\partial x^{(s-1)}} = \frac{\partial z^{(s)}}{\partial x^{(s-1)}}\frac{\partial x^{(s)}}{\partial z^{(s)}}\frac{\partial C_0}{\partial x^{(s)}} \tag{3.13.}$$

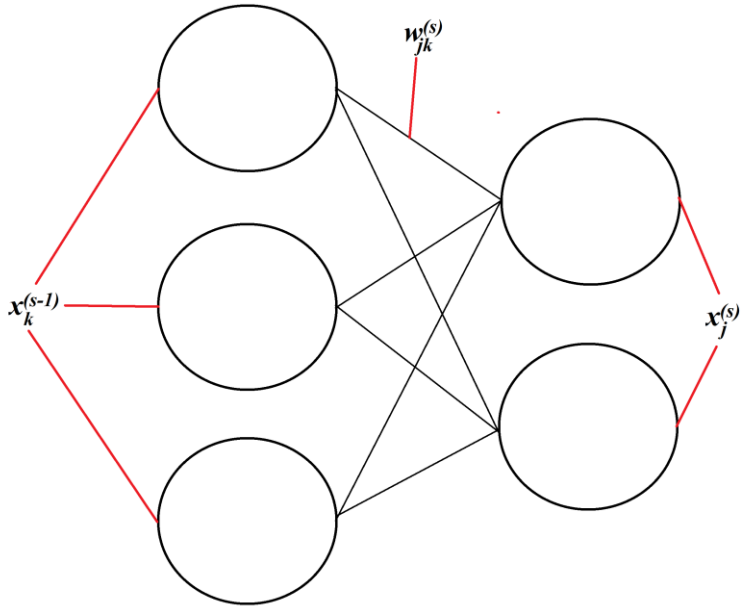
Iako su za izvođenje navedenih jednadžbi uvedena pojednostavljenja u obliku smanjenog broja slojeva i neurona po istima, mreže s više neurona po sloju imaju suštinski iste jednadžbe izuzev malih izmjena. Kao primjer jednadžba promjene funkcije cilja po težinama glasi:

$$\frac{\partial C_0}{\partial w_{jk}^{(s)}} = \frac{\partial z_j^{(s)}}{\partial w_{jk}^{(s)}}\frac{\partial x_j^{(s)}}{\partial z_j^{(s)}}\frac{\partial C_0}{\partial x_j^{(s)}} \tag{3.14.}$$

Iz priloženog je vidljivo da je jedina razlika u indeksima, gdje *k* označava poziciju neurona u prethodnom sloju mreže, a *j* poziciju u sljedećem sloju. Jednadžba promjene po težinama Bias neurona ima iste modifikacije, dok jednadžbe promjene po izlazu iz prijašnjeg sloja sadrži dodatnu razliku:

$$\frac{\partial C_0}{\partial x_k^{(s-1)}} = \sum_{j=0}^{n_s-1}\frac{\partial z_j^{(s)}}{\partial x_k^{(s-1)}}\frac{\partial x_k^{(s-1)}}{\partial z_j^{(s)}}\frac{\partial C_0}{\partial x_j^{(s)}} \tag{3.15.}$$

U jednadžbu je potrebno uvesti sumu jer neuron iz prijašnjeg sloja utječe na sve neurone u slijednom sloju, pa je time potrebno u obzir uzeti čitav utjecaj koji isti ima na sljedeći sloj. Radi lakšeg shvaćanja varijabli i indeksa istih, slika 3.10. prikazuje mrežu s više slojeva i neurona nego prijašnja.



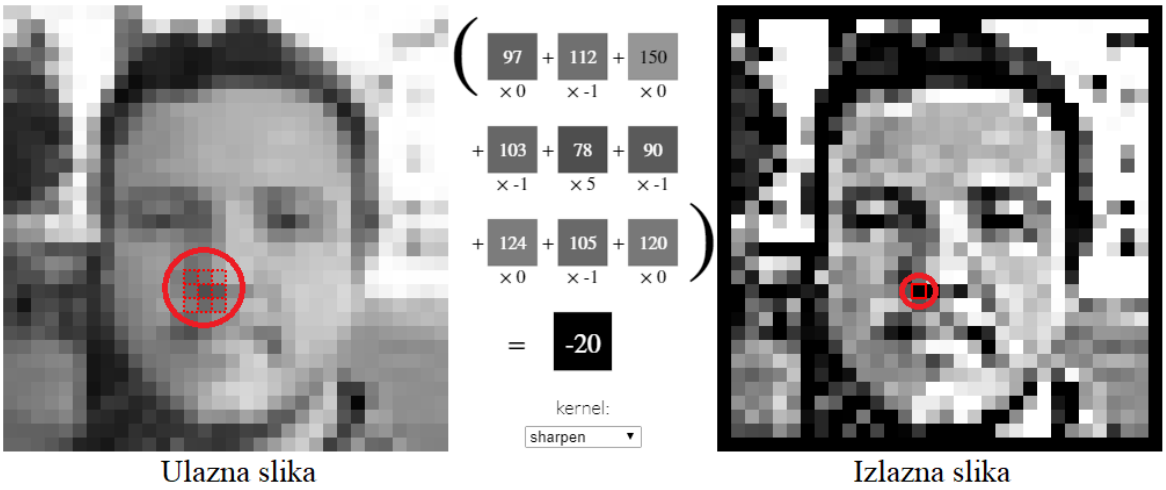
Slika 3.10. Jednostavna neuronska mreža sa 2 sloja i 5 neurona

3.4. Umjetne konvolucijske neuronske mreže

Poput umjetnih neuronskih mreža obrađenih u prijašnjem podnaslovu, i umjetne konvolucijske mreže su modelirane po uzoru na prirodu, točnije prema vidnom korteksu sisavaca. Znanstvenici Hubel i Wiesel su sredinom 20. stoljeća otkrili da neuroni smješteni u vidnom korteksu majmuna i mačaka zasebno reagiraju na podražaje u malim dijelovima vidnog polja. To znači da se mala skupina neurona fokusira na mali dio vidnog polja, te povezujući i preklapajući više takvih skupina mozak stvara kompletnu sliku. Prva implementacija navedenog je izvedena 1989. u svrhu prepoznavanju rukom pisanih znamenki. Autor te implementacije je Yann LeCun, te se korištena mreža naziva LeNet-5 i ujedno je prva izvedba konvolucijske neuronske mreže na računalu [7].

Takva struktura mreže se pokazala znatno kvalitetnijom u području prepoznavanja teksta i govora, te predstavlja nadogradnju na klasičnu strukturu neuronskih mreža. Osnovna razlika običnih neuronskih mreža i konvolucijskih je u tome da slojevi nisu „gusto“ povezani, što znači da svaki neuron prethodnog sloja nije nužno povezan se svakim neuronom u slijednom sloju. Kao kod bioloških sustava povezane su samo male skupine neurona te one kao takve tvore lokalna receptivna polja. Na području prepoznavanja teksta prednost takve strukture je da može raditi sa slikama puno veće rezolucije, jer je broj parametara po jednakoj veličini mreže znatno manji. Osim toga mreža je znatno robusnija na transformacije, promjene ulaznog signala te šum.

Princip rada konvolucijskog sloja takve neuronske mreže prikazan je na slici 3.11.



Slika 3.11. Princip rada konvolucijskog sloja umjetne neuronske mreže[8]

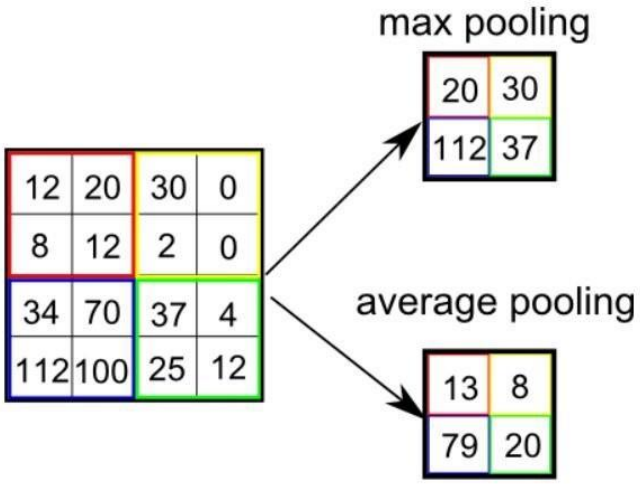
Na lijevoj strani je prikazana slika koja je u stvari ulazni sloj mreže, te se na njoj nalazi mreža kvadratića dimenzija 3x3 poznata kao filter. Pomicanjem filtera se odabiru polja odnosno neuroni koji se množe s vrijednostima težina prikazanim na sredini slike. Konačno se pomnožene vrijednosti zbrajaju i izlazna vrijednost je osvijetljenost piksela u izlaznoj slici. Filter prikazan na slici 3.11. sadrži unaprijed određene vrijednosti težina koje iznose: 0 na rubovima, -1 na stranicama i 5 u sredini, te je njegov zadatak izoštravanje slike. U procesu učenja mreža počinje s nasumičnim vrijednostima težina *w* te ih mijenja tako da pojedini filteri postaju specijalizirani za prepoznavanje specifičnih značajki na slici, pa će tako neki od filtera biti specijalizirani za pronalaženje rubova, dok će drugi prepoznavati linije ili nekakve posebne značajke. Veličina filtera i koraka kojom se filtera pomiče su parametari konvolucijske neuronske mreže koji u konačnici utječu na performanse. Za primjer, manja vrijednost koraka osigurava više preklapanja i izlaznu sliku veće rezolucije, a time i veću točnost mreže. Postupak konvolucije se može zapisati matematički:

$$z_{i,j} = \sum_{k=1}^{m} \sum_{l=1}^{m} w_{k,l} x_{i+k-1,j+l-1} \tag{3.16.}$$

Oznaka *z* predstavlja izlaz neurona konvolucijskog sloja, indeksi *i* i *j* označavaju sloj i mjesto neurona u istom, a oznaka *m* dimenzije filtera.

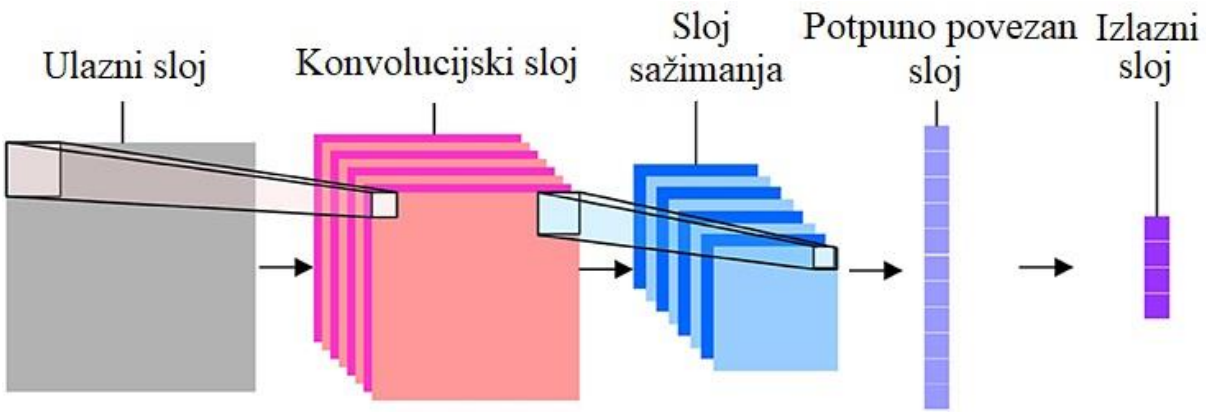
Jedno od bitnih obilježja konvolucijskih neuronskih mreže je slijedno slaganje različitih slojeva, pa tako se osim konvolucijskog i običnog gusto povezanog sloja, često nalazi i sloj

sažimanja (*eng. Pooling layer*). Takav sloj smanjuje količinu informaciju sadržanu u slici na način da iz skupa podataka odabire najveći (*eng. max pooling*) ili uzima prosječnu vrijednost podataka (*eng. average pooling*) te je prosljeđuje u sljedeći sloj. Na taj način se znatno smanjuju potrebni resursi potrebni za treniranje mreže, kao primjer koristeći sloj sažimanja s filterom dimenzija 2x2 i korakom 2, iz slike se uklanja 75% informacija. Opisani postupak sažimanja prikazan je na slici 3.12.



Slika 3.12. Vizualni prikaz rada sloja sažimanja

Koristeći kombinacije 3 sloja navedena u ovom podnaslovu se kreiraju duboke konvolucijske mreže, i primjer jedne takve mreže prikazan je na slici 3.13. Prikazana struktura se sastoji se od konvolucijskog sloja, sloja sažimanja te potpuno povezanog sloja. Ovisno o vrsti zadatka mreža može imati i više konvolucijskih slojeva, slojeva sažimanja i potpuno povezanih slojeva.



Slika 3.13. Pojednostavljena struktura konvolucijske mreže

4. PROGRAMSKA IMPLEMENTACIJA NEURONSKIH MREŽA

U ovom poglavlju će biti analizirani i uspoređeni rezultati prepoznavanja rukom pisanih brojki i slova koristeći umjetne neuronske mreže. Početni dio poglavlja posvećen je upoznavanju s programskim jezikom Python i pripadajućim programskim bibliotekama TensorFlow, NumPy i Matplotlib. Također će biti prikazani setovi podataka za učenje MNIST i EMNIST te njihove karakteristike. Kasniji dio poglavlja će obraditi rezultate dobivene koristeći navedenu podršku te usporediti performanse različitih tipova mreža uz različite vrijednosti parametara.

4.1. Python, TensorFlow i pomoćne biblioteke

Python je programski jezik visoke razine i opće namjene stvoren 1990. godine. U Pythonu je moguće koristiti različite stilove programiranja pa tako zavisno preferencijama, programer može odabrati objektno, strukturno ili aspektno orijentirano programiranje. Upravo takva fleksibilnost i licenca otvorenog koda je pridonijela rastućoj popularnosti Pythona u industrijskom i komercijalnom okruženju. Osim spomenutog, za navedeni jezik se mogu preuzeti službene kao i biblioteke kreirane od strane korisnika koje znatno olakšavaju rješavanje nebrojenih problema[9].

Pa je tako i tvrtka Google kreirala besplatnu biblioteku TensorFlow za korištenje u različitim programskim jezicima među kojima je i Python. TensorFlow znatno olakšava izradu, treniranje i testiranje modela strojnog učenja, i na taj način omogućava pojedincima slabijih programerskih vještina kao i profesionalcima lakše korištenje ove inovativne tehnologije. Kao što je vidjivo iz programske podrške, jednostavne neuronske mreže se mogu kreirati s manje od 100 linija koda. Zahvaljujući ovakvom pristupu sve više kompanija počinje koristiti umjetnu inteligenciju, pa se tako nalaze rješenja za probleme koji su se do sad smatrali nemogućima.

Osim TensorFlow-a, u izradi programskog koda korištena je i biblioteka NumPy. U njoj se nalaze esencijalne matematičke funkcije korištene u linearnoj algebri, razne transformacije, mogućnost stvaranja nasumičnih vrijednosti te naravno mogućnost rada s matricama.

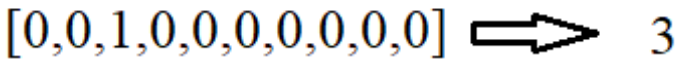
Uz NumPy, važno je spomenuti i biblioteku Matplotlib, koja omogućava jednostavnu vizualizaciju podataka. Modul *pyplot* iz spomenute biblioteke je korišten kod prikazivanja metrika učenja mreža korištenih u programskoj podršci.

4.2. MNIST i EMNIST baza podataka

Baze podataka korištene u ovom radu za učenje i treniranje umjetnih neuronskih mreža su MNIST (*eng. Modified National Institute of Standards and Technology*) i EMNIST (*eng. Extended Modified National Institute of Standards and Technology*). Razlika tih dviju baza podataka je ta da EMNIST sadrži skup slika rukom pisanih velikih i malih slova te brojki, dok MNIST sadrži samo rukom pisane brojke. Dodatna razlika je u količini podataka koje sadrže, MNIST baza podataka sadrži 60,000 slika namijenjenih za učenje mreže, a 10,000 za testiranje, dok su te brojke za EMNIST znatno više i iznose 697,932 slika za učenje mreže i 116,323 za testiranje. Svaki od njih se sastoji od seta slika u sivim tonovima dimenzija 28x28 piksela, drugim riječima slike su predobrađene što znatno poboljšava točnost modela. Primjeri podataka sadržanih u spomenutim bazama podataka prikazani su na slici 4.1.

Slika 4.1. Primjeri iz EMNIST baze podataka

Uz to svaka slika sadrži oznaku (*eng. label*) koja daje stvarnu vrijednost brojke prikazane na slici, te se koristeći *One-hot* kodiranje dobiva povoljniji format za ocjenjivanje modela. *One-hot* kodiranje je algoritam koji zapisuje oznake kao vektore binarnih vrijednosti, te za prepoznavanje brojki funkcionira tako da je vrijednost oznake sadržana u vekoru sa 10 stupaca, pa ovisno koja je brojka prikazana na slici na odgovarajućem mjestu u vektoru se nalazi vrijednost 1, a na ostalima vrijednosti 0. Primjer navedenog je prikazan na slici 4.2.



Slika 4.2. Primjer *One-hot* kodiranja

Kod prepoznavanja brojki i slova, *One-hot* kodiranje funkcionira na isti način samo je vektor znatno veći i sadrži 62 binarne vrijednosti.

4.3. Rezultati učenja standardne neuronske mreže na problemu prepoznavanja brojki

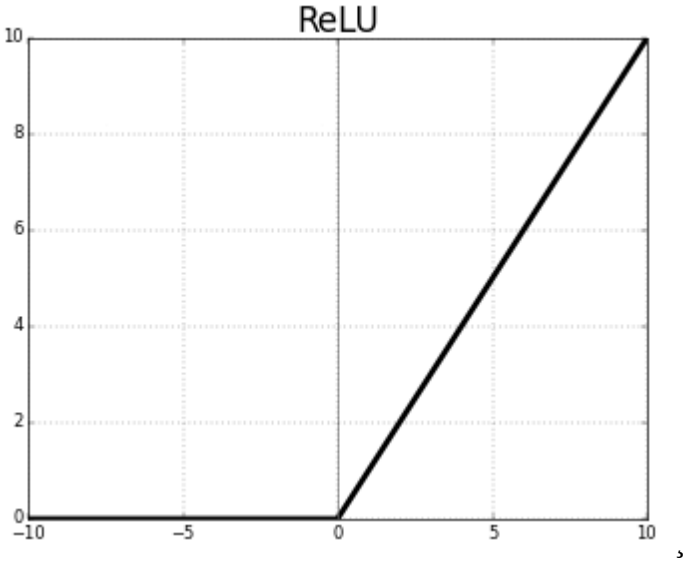
U ovom podnaslovu će biti prikazano učenje mreže na temelju greške koja mreža daje pri prepoznavanju rukom pisanih brojki. Prezentirati i usporediti će se rezultati učenja mreže zavisno o parametrima učenja poput:

- koeficijenta brzine učenja
- broja koraka učenja
- broju skrivenih slojeva mreže
- broj neurona po sloju
- vrsti algortima optimizacije

Za početne težine i pragove su uzete nasumične vrijednosti iz normalne distribucije sa standardnom devijacijom iznosa 0.1. Odabrana funkcija cilja je softmax funkcija ukrštene entropije, dok je za aktivacijsku funkciju korištena ReLu (*eng. rectified linear unit*) funkcija. Ta aktivacijska funkcija je odabrana zbog superiornih performansi u odnosu na sigmoidnu i aktivacijsku funkcija tangensa hiperboličnog. Matematički zapis aktivacijske funkcije opisane glasi:

$$f(x) = \begin{cases} 0 \text{ za } x < 0 \\ x \text{ za } x \geq 0 \end{cases} \tag{4.1}$$

Odnosno za ulazne vrijednosti manje od 0, izlazna vrijednost funkcije je 0, dok je za ulazne vrijednosti veće od 0, izlazna vrijednost funkcije jednaka ulaznoj. Prikaz spomenute funkcije se nalazi na slici 4.3.



Slika 4.3. ReLu aktivacijska funkcija

Vrijednost serije podataka (*eng. batch*) određuje koliko primjera za učenje mreža uzima tijekom učenja. Veći broj primjera karakterizira veća kvaliteta učenja, te su u ovom slučaju odabrane serije od 50 primjera, što znači da mreža svakih 50 slika mreža podešava težine i pragove.

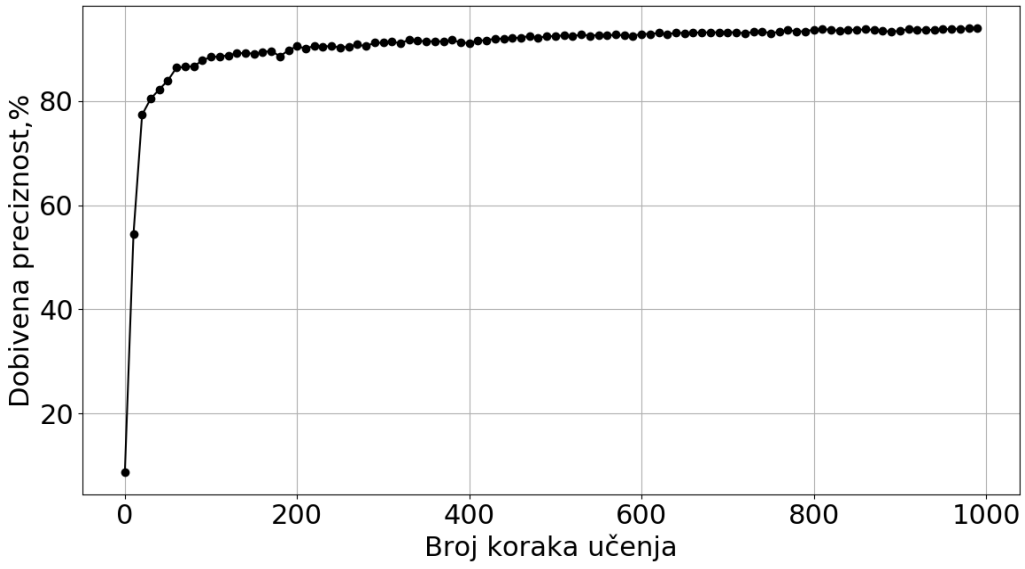
Pri učenju mreže svakih 10 koraka je izmjerena je i spremljena točnost mreže. Na temelju tih vrijednosti su ispisani grafički prikazi ovisnosti točnosti mreže o broju koraka učenja. U nastavku su prikazani slučajevi sa različitim parametrima, te odgovarajući prikazi učenja mreže.

Prvi slučaj

Za prvi slučaj učenja uzeta je poprilično jednostavna neuronska mreža sa 1 skrivenim slojem koji sadržava 300 neurona. Koeficijent brzine učenja je malo veći od standarnog, dok je za algoritma optimizacije odabran gradijentni spust. Ista mreža je testirana sa različitim brojem koraka u iznosima od 1000, 3000 i 5000. Tablica 4.1. prikazuje odabrane parametre pri prvom testiranju mreže.

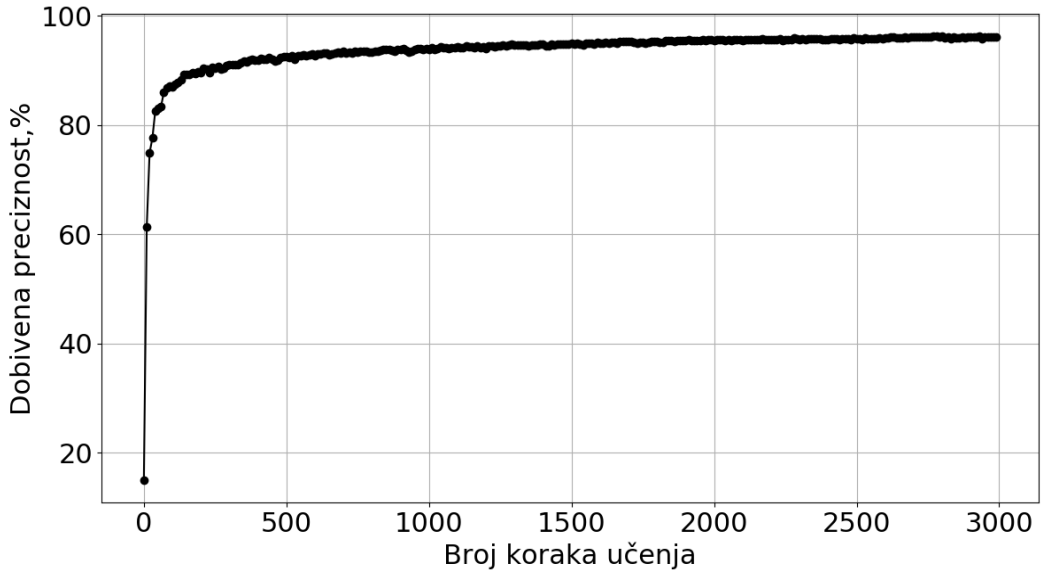
Tablica 4.1. Parametri mreže za prvi slučaj učenja

Koeficijent brzine učenja	0.1
Broj koraka učenja	1000, 3000, 5000
Broj skrivenih slojeva mreže	1
Broj neurona	300
Vrsta algoritma optimizacije	Algoritam gradijentnog spusta



Slika 4.4. Ovisnost točnosti o koraku učenja za prvi slučaj učenja sa 1000 koraka

Slika 4.4. prikazuje način na koji se povećava točnost mreže pri 1000 koraka učenja, te u konačnici mreža postiže grešku od 93.96%, što je i više nego prihvatljivo za ovako jednostavnu strukturu. Ista mreža s većim brojem koraka je prikazana na slici 4.5, te je jasno vidljivo da se povećanjem broja koraka povećava i točnost mreže. Dodatnim povećanjem koraka na 5000, ne dolazi do znatnog povećanja točnosti, pa će se ostali slučajevi učenja fokusirati upravo na mrežu sa 3000 koraka učenja.



Slika 4.5. Ovisnost točnosti o koraku učenja za prvi slučaj učenja sa 3000 koraka

Iz grafičkih prikaza mreže očito je da mreža najviše „nauči“ u prvih 300 koraka, dok se u ostatku učenja točnost minimalno poveća.

Tablica 4.2. Broj koraka i rezultirajuća točnost za prvi slučaj učenja

Broj koraka	1000	3000	5000
Točnost učenja	93.96%	96.13%	96.87%

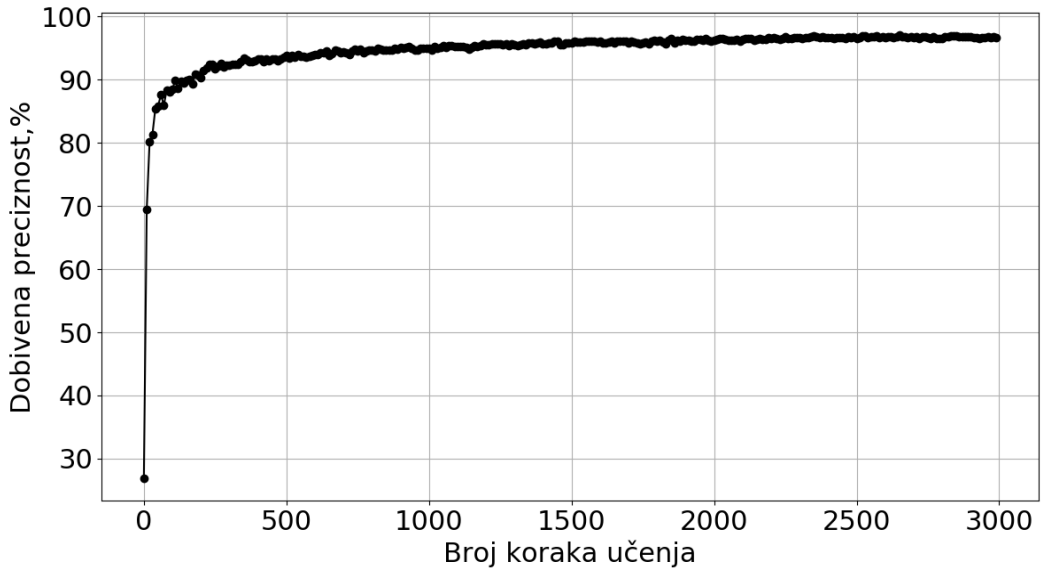
Promjenom koeficijenta brzine učenja se također ne dobivaju značajna poboljšanja, veći iznos daje iste pa čak i niže točnosti, dok kod nižeg iznosa mreža ne uspijeva dostići minimum funkcije cilja za navedene brojeve koraka. Pošto se pomoću tih varijabi ne može dobiti veća točnost, uvodi se dodatan sloj sa 150 neurona, te u se u prvi sloj dodaju neuroni tako da ukupna količina u njemu iznos 500 neurona.

Drugi slučaj

Tablica 4.3. Parametri mreže za drugi slučaj učenja

Koeficijent brzine učenja	0.1
Broj koraka učenja	3000
Broj skrivenih slojeva mreže	2
Broj neurona	500 + 150
Vrsta algoritma optimizacije	Algoritam gradijentnog spusta

Koristeći poboljšana u tablici 4.3, vrijeme treniranja mreže se povećava, ali se zato i točnost povećava za gotovo 1% te u ovom slučaju iznosi 97.03%. Postupak učenja prikazan je na slici 4.6.



Slika 4.6. Ovisnost točnosti o koraku učenja za slučaj učenja sa 2 skrivena sloja

Iako bi se teoretski točnost mogla dalje povećati uvođenjem dodatnih poboljšanja, poput dodavanja većeg broja slojeva i neurona u istima, to bi zahtjevalo znatno više računalnih resursa nego je dostupno. Iz toga razloga se ta dodatna poboljšana neće razmatrati, već će se u nastavku usporediti rezultati mreže za algoritam gradijentnog spusta s koeficijent momentuma.

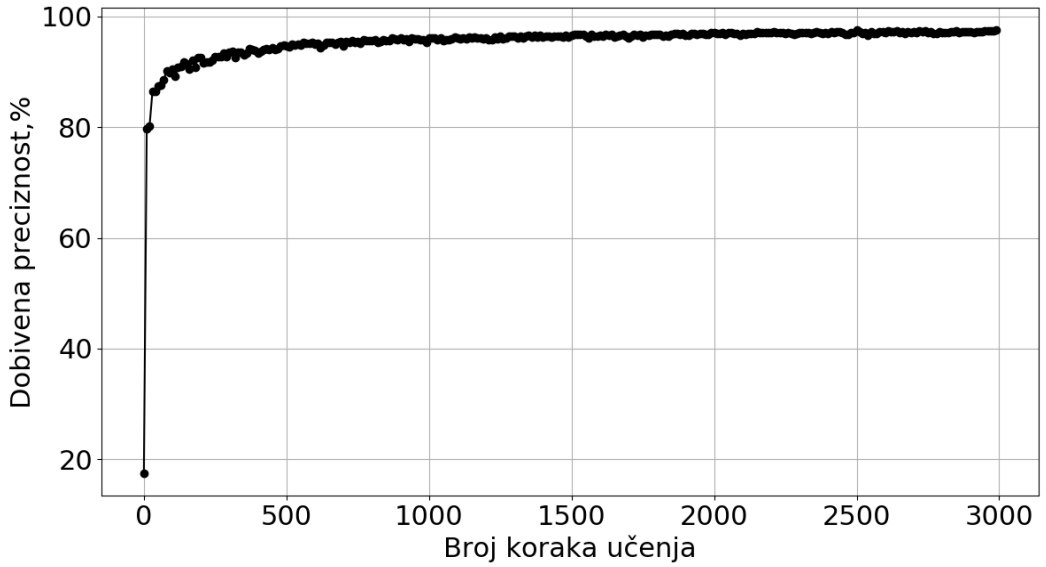
Treći slučaj

Treći slučaj je uveden radi prikaza razlike točnosti mreže pri dodavanju koeficijenta momentuma objašnjenog u prijašnjem poglavlju. Ovdje će biti prikazani slučajevi sa više različitih vrijednosti momentuma uz nepromijenjene ostale parametre mreže.

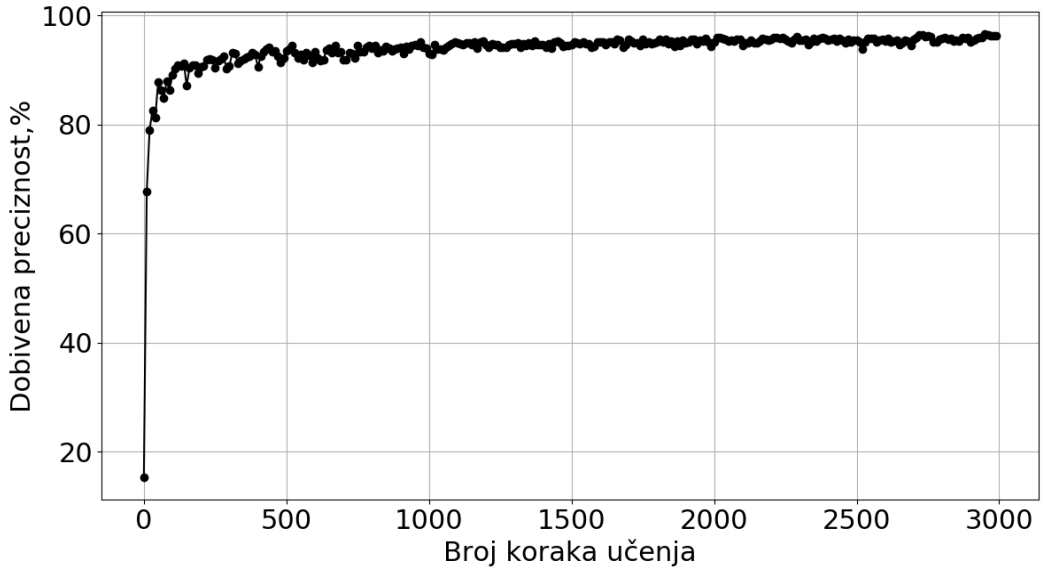
Tablica 4.4. Parametri mreže za drugi slučaj učenja

Koeficijent brzine učenja	0.1
Broj koraka učenja	3000
Broj skrivenih slojeva mreže	2
Broj neurona	500+150
Vrsta algoritma optimizacije	Algoritam gradijentnog spusta sa momentumom

U nastavku, na slikama 4.7. i 4.8. su prikazane točnosti učenja ovisno o broju koraka iz kojih je jasno se vidi da mreža sa koeficijentom momentuma iznosa 0.5 brže minimizira grešku, ali uz veće perturbacije.



Slika 4.7. Ovisnost točnosti o koraku učenja za slučaj učenja sa iznosom momentuma 0.5



Slika 4.8. Ovisnost točnosti o koraku učenja za slučaj učenja sa iznosom momentuma 0.9

Rezultati učenja prikazani su u tablici 4.5, te je očito da srednja vrijednost momentuma u iznosu od 0.5 rezultira najvećom točnošću mreže.

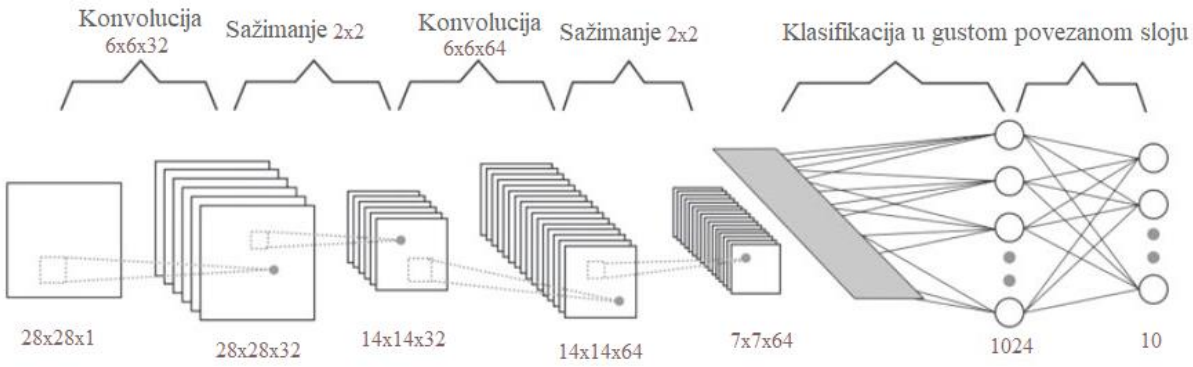
Tablica 4.5. Iznos momentuma i rezultirajuća točnost za treći slučaj učenja

Iznos koeficijenta momentuma	0.9	0.5	0.1
Točnost učenja	96.58%	97.52%	97.20%

4.4. Rezultati učenja konvolucijske neuronske mreže na problemu prepoznavanja brojki

Za razliku od klasičnih neuronskih mreža, konvolucijske neuronske mreže zahtjevaju velike količine računalnih resursa, a vremena treniranja postaju sve duža. Iz toga razloga napravljene su samo tri varijacije i to u koeficijentu brzine učenja, te su njihovi prikazi i odgovarajuće greške dane u tablici 4.6. Kao i kod običnih neuronskih mreža, težine i pragovi su nasumično inicijalizirani, te je korištena ista funkcija cilja i jednak broj koraka učenja. Veličine serije su također ostale iste, te su na istim koracima uzete vrijednosti točnosti. Jedina osobita razlika je u korištenju algoritma optimizacije, dok je u prijašnjem primjeru korišten gradijentni spust sa i bez koefcijenta momentuma, ovdje je korišten „Adam optimizator“.

Struktura korištene mreže mreže prikazana je na slici 4.9. i sastoji se od 2 konvolucijska sloja, 2 sloja sažimanja te jednog gusto povezanog sloja.



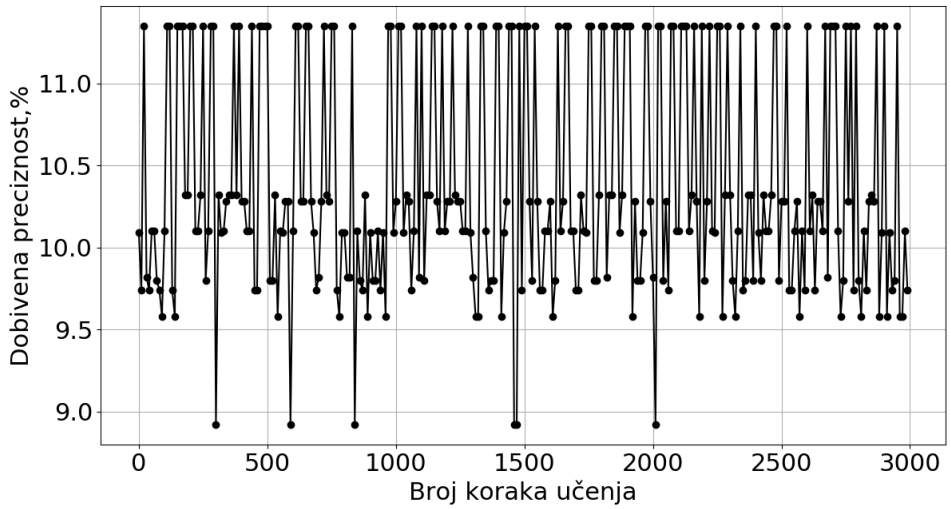
Slika 4.9. Struktura konvolucijske neuronske mreže

Za veličinu filtera odabrana je vrijednost 6x6, te je korak konvolucije 1. Za sažimanje je korišten algoritam maksimalnog sažimanja dimenzija 2x2.

Tablica 4.6. Iznos koeficijenta brzine učenja i rezultirajuća točnost

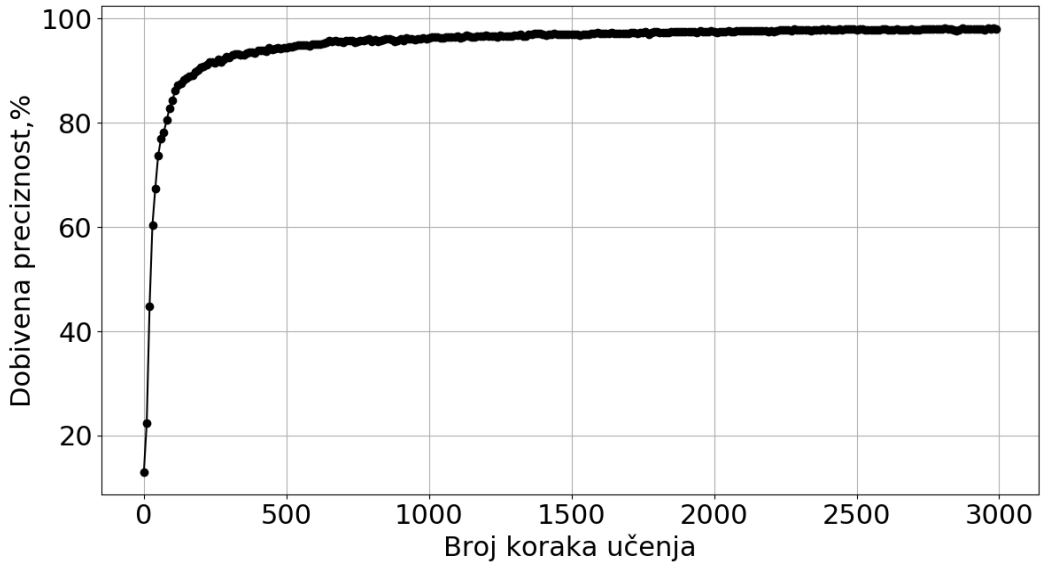
Koeficijent brzine učenja	0.1	0.01	0.0001
Točnost mreže	9.74%	96.64%	98.07%

Iz tablice je očito da uz pravu vrijednost koeficijenta brzine učenja, konvolucijska neuronska mreža daje točnosti iznad 98% za problem prepoznavanja rukom pisanih brojeva.



Slika 4.10. Ovisnost točnosti o koraku učenja konvolucijske neuronske mreže sa koeficijentom brzine učenja 0.1

Slika 4.10. prikazuje slučaj učenja gdje koeficijent brzine učenja prevelik, te izlazi funkcije cilja divergiraju. Posljedica toga je mreža s vrlo lošim performansama i perturbacijama točnosti. Sljedeća slika 4.11. prikazuje savršen primjer učenja mreže, sa odlično odabranim parametrima gdje se svakim korakom učenja povećava točnost mreže i skoro dostiže maksimalni postotak iste.



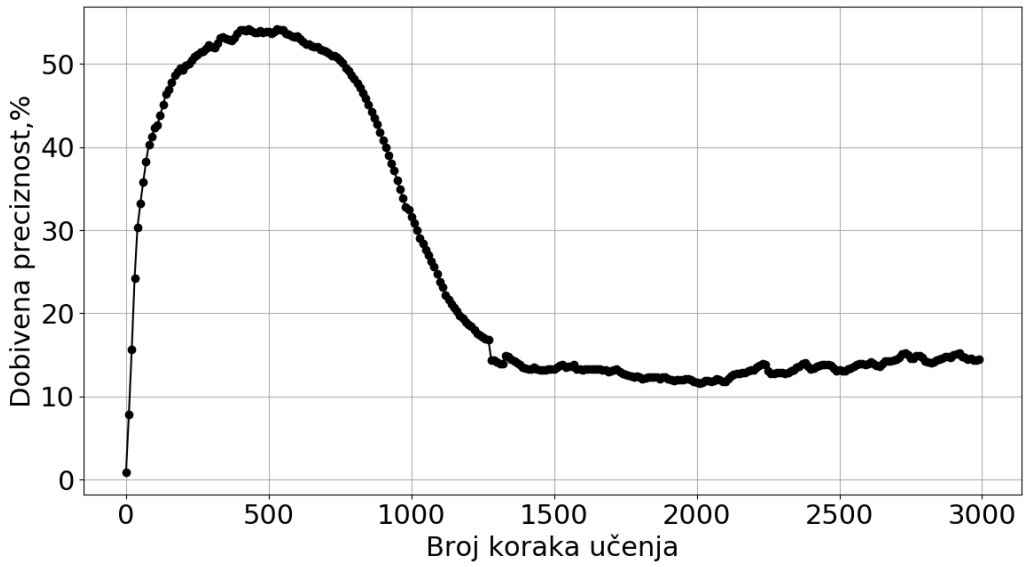
Slika 4.11. Ovisnost točnosti o koraku učenja konvolucijske neuronske mreže sa koeficijentom brzine učenja 0.0001

Slika 4.12. prikazuje testne slike brojki na kojima je mreža napravila grešku, te iznad njih predviđene i stvarne vrijednosti istih. Iz prikazanog je očito da mreže nisu savršene, te iako mogu imati visoke postotke točnosti i dalje rade grube greške na jednostavnim primjerima.

Slika 4.12. Prikazi brojki na kojima je mreža napravila pogrešku

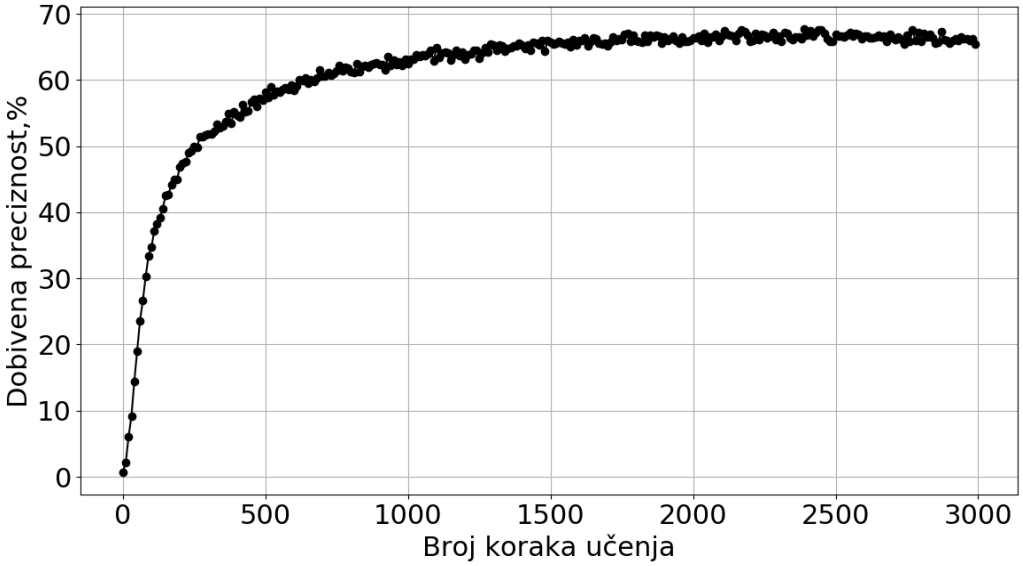
4.5. Rezultati učenja konvolucijske neuronske mreže na problemu prepoznavanja rukom pisanih brojki i slova

Ovaj podnaslov će također obraditi korištenje konvolucijskih neuronskih mreža, no razlika u odnosu na prijašnji je ta da je ovdje korištena baza podataka znatno kompleksnija. Riječ je naravno o EMNIST bazi podataka spomenutoj na početku poglavlja, koja se sastoji od 62 klase, gdje je 26 klasa korišteno za velika slova, 26 klasa za mala slova, te 10 klasa za brojeve. Kao dokaz kompleksnosti ovog problema, koristiti će se konvolucijska mreža iz prijašnjeg podnaslova, uz iste parametre učenja. Točnost tijekom učenja prikazana je na slici 4.13., te je očito da takva mreža nije dorasla zadatku.



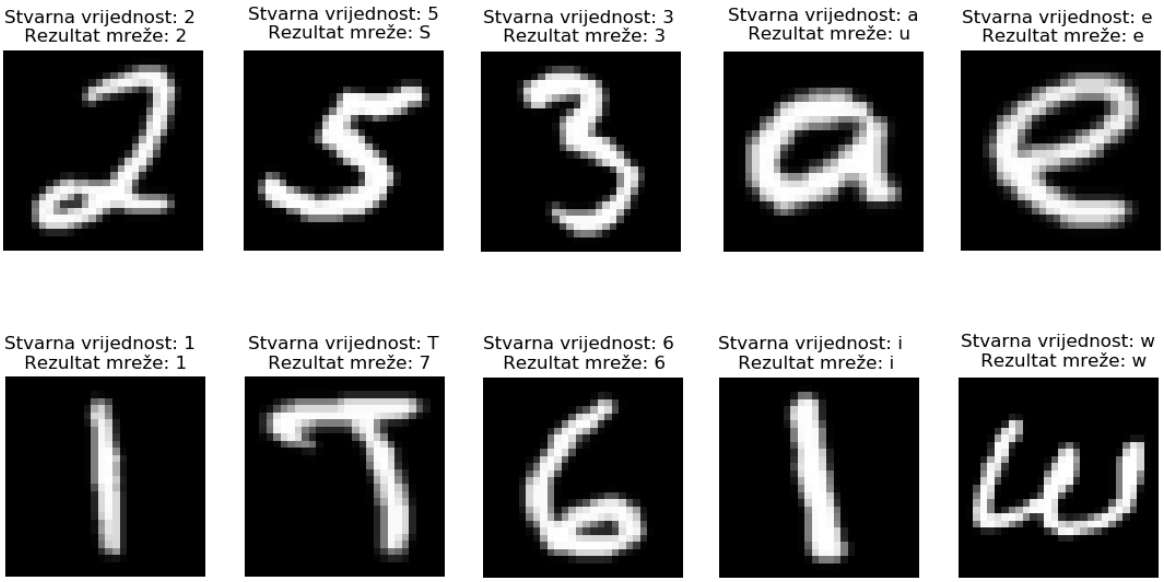
Slika 4.13. Ovisnost točnosti o koraku učenja konvolucijske neuronske mreže sa EMNIST bazom podataka

Maksimalna postignuta točnost iznosi 54.17%, nakon koje mreža stagnira te joj konačno točnost opada na 14.46%. Uz modifikacije skrivenih neurona i parametara mreže se mogu postići i veće točnosti, pa je tako veličina filtera smanjena na dimenzije 3x3, broj neurona u konvolucijskim slojevima je povećan na 128, a u potpuno povezanom sloju na 2048, dok je vrijednost koraka učenja smanjena na 0.00003. Na slici 4.14. je prikazana projmena točnosti učenja za tako modificiranu mreži i očito je da ni za više koraka učenja mreža neće dati bolje rezultate.



Slika 4.14. Ovisnost točnosti o koraku učenja modificirane konvolucijske neuronske mreže sa EMNIST bazom podataka

Zbog nedostatka računalnih resursa, dodatna poboljšana nažalost neće biti razmatrana pa tako konačna točnost mreže sa ovaj slučaj učenja iznosi 65.49%. Kao finalni test, mreži je nakon 3000 koraka dano 10 nasumičnih primjera, te su na slici 4.15. prikazani njeni odzivi i stvarne vrijednosti slika. Iako je u ovom slučaju mreža dala točan rezultat u 70% slučaja, korištenjem većeg broja uzoraka, taj broj konvergirao prema dobivenoj točnosti od 65.49%.



Slika 4.15. Testiranje mreže na 10 nasumičnih primjera

5. Zaključak

Unutar završnog rada prikazane su strukture suvremenih neuronskih i konvolucijskih neuronskih mreža pri rješavanju problema prepoznavanja rukom pisanih brojki i slova. Modeli kreirani u programkom paketu Python uz pomoć biblioteke TensorFlow, su trenirani i testirani na preuzetim bazama podataka MNIST i EMNIST. Različitim kombinacijama parametara i struktura dan je prikaz ovisnosti performansi mreža o istima. Pokazano je da relativno jednostavne neuronske mreže, kreirane u kratkom vremenskom roku mogu imati jako velike točnosti u slučaju prepoznavanja brojki te dostižu rezultate gotovo slične ljudskima. Također je prikazan razmjer rasta kompleksnosti strukture mreže u slučaju povećanja kompleksnoti problema koji se pokušava riješiti, pa tako mreža koja savršeno prepoznaje brojke ne daje zadovoljavajuće rezultate pri prepoznavanju brojki i slova. Daljnja poboljšanja izuzev povećanja broja neurona i skrivenih slojeva, promjena veličina filtera i koraka učenja, mogu biti implementirana u obliku predobrade podataka te korištenjem efektivnijih optimizatora.

Zbog kompleksnosti problema koje umjetne neuronske mreže danas rješavaju, često prevladava mišljenje da je to područje rezerevirano isključivo za pojedince sa programerskom pozadinom. Pojavom raznih programskih bibiloteka, široj javnosti je omogućeno korištenje algoritama neuronskih mreža što dodatno doprinosi njihovoj popularizaciji i time postaju sve korisnijim alatom za rješavanje problema bilo to u istraživačke ili privatne svrhe.

LITERATURA

[1] https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html

[2] https://medium.com/swlh/applications-of-ocr-you-havent-thought-of-69a6a559874b

[3] https://www.ijcaonline.org/research/volume135/number2/manisha-2016-ijca-908349.pdf

[4] https://medium.com/@himanshubeniwal/handwritten-digit-recognition-using-machine-learning-ad30562a9b64

[5] http://yann.lecun.com/exdb/mnist/

[6] https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

[7] https://en.wikipedia.org/wiki/Convolutional_neural_network

[8] http://setosa.io/ev/image-kernels/

[9] https://en.wikipedia.org/wiki/Python_(programming_language)