**SVEUČILIŠTE U SPLITU**

**FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE**

**ZAVRŠNI RAD**

**RAZVOJ I ANALIZA KONVOLUCIJSKIH NEURONSKIH MREŽA**

**Roko Dumanić**

**Split, rujan 2020.**

SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Preddiplomski sveučilišni studij: **Računarstvo**

Oznaka programa: 120

Akademska godina: 2019/2020

Ime i prezime: **Roko Dumanić**

Broj indeksa: 264-2016

**ZADATAK ZAVRŠNOG RADA**

Naslov: **Razvoj i analiza konvolucijskih neuronskih mreža**

Zadatak: Dati pregled metoda dubinskog učenja s naglaskom na konvolucijske neuronske mreže (CNN). Upoznati se sa razvojem i analizom algoritama temeljenih na konvolucijskim neuralnim mrežama kroz praktični primjer prepoznavanja znamenki primjenom CNN arhitekture po izboru na MNIST slikovoj bazi u TensorFlow okruženju.

Datum obrane: 23. 09. 2020.

Mentor:

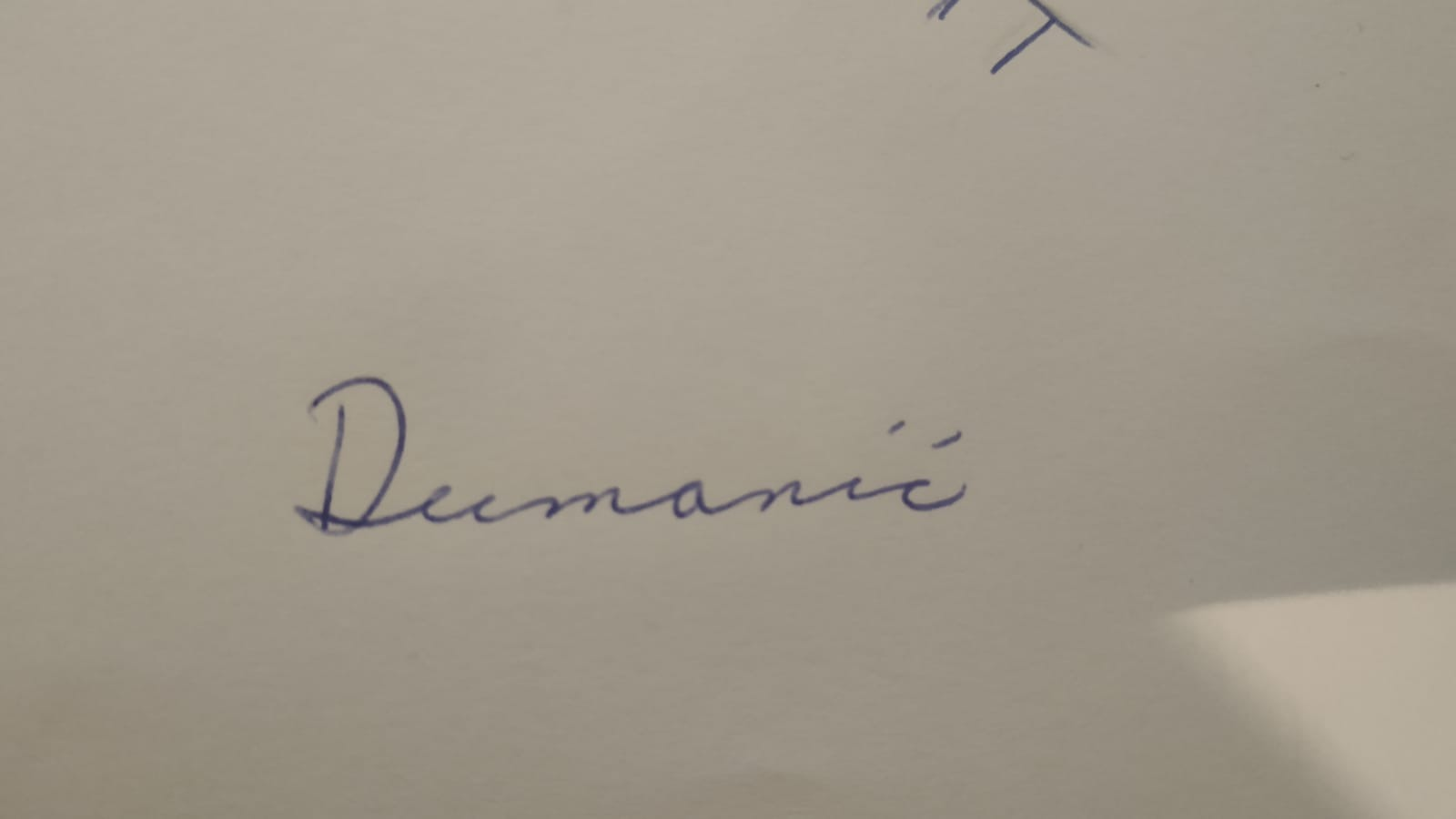


doc. dr. sc. Ana Kuzmanić Skelin

**IZJAVA**

Ovom izjavom potvrđujem da sam završni rad s naslovom „Razvoj i analiza konvolucijskih neuronskih mreža“ pod mentorstvom doc. dr. sc. Ana Kuzmanić Skelin pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo u završnom radu citirao sam i povezao s korištenim bibliografskim jedinicama.

Student



Roko Dumanić

# Sadržaj

[1 UVOD 1](#_Toc51158550)

[2 TENSORFLOW 2](#_Toc51158551)

[2.1 Podaci i operacije 2](#_Toc51158552)

[2.2 Kod kao graf 2](#_Toc51158553)

[2.3 Vrste 3](#_Toc51158554)

[2.4 Jupyter notebook 3](#_Toc51158555)

[3 Google Colaboratory 5](#_Toc51158556)

[3.1 Organizacija koda 5](#_Toc51158557)

[3.2 Hardver 5](#_Toc51158558)

[4 Strojno učenje 7](#_Toc51158559)

[4.1 Skup podataka 7](#_Toc51158560)

[4.2 Neuronske mreže 7](#_Toc51158561)

[4.3 Konvolucijske neuronske mreže(CNN) 8](#_Toc51158562)

[4.4 Težine neurona 8](#_Toc51158563)

[4.5 Propagacija 9](#_Toc51158564)

[4.6 Model mreže 9](#_Toc51158565)

[4.7 Sekvencijalni model 10](#_Toc51158566)

[4.8 Redukcija dimenzije 10](#_Toc51158567)

[4.9 Sažimanje 10](#_Toc51158568)

[4.10 Algoritam gradijentnog spusta 11](#_Toc51158569)

[4.11 Prevođenje modela na strojni jezik 11](#_Toc51158570)

[4.12 Podešavanje modela 11](#_Toc51158571)

[5 Optimizacija 13](#_Toc51158572)

[5.1 Adam 13](#_Toc51158573)

[5.2 Adamax 17](#_Toc51158574)

[5.3 RMSprop 19](#_Toc51158575)

[5.4 SGD 21](#_Toc51158576)

[Zaključak 23](#_Toc51158577)

[Literatura 24](#_Toc51158578)

[Popis oznaka i kratica 25](#_Toc51158579)

[Sažetak 26](#_Toc51158580)

[Ključne riječi 26](#_Toc51158581)

[Summary 27](#_Toc51158582)

[Keywords 27](#_Toc51158583)

[Title 27](#_Toc51158584)

1. UVOD

Strojno učenje izgleda kao najnovija tehnologija razvijena nedavno, ali zapravo s postepeno razvija više od pola stoljeća. Navest ćemo nekoliko događaja koji su obilježili početak razvoja.

Alan Turing je 1950. smislio „Turingov test“ kako bi provjerio da li računalo ima pravu inteligenciju. U testu čovjek i računalo pričaju, ako računalo može uvjeriti čovjeka da je i ono čovjek tada je prošlo test. [[1](#_Literatura)]

1950-ih Arthur Lee Samuel je napisao prvi program sa sposobnosti učenja. Program je bio igra dame i što je više igrao postajao je bolji.[[1](#_Literatura)][[2](#_Zaključak)]

Prvu neuronsku mrežu je napravio Frank Rosenblatt 1957. Zvala se „The Perceptron“ i simulirala je tok razmišljanja ljudskog mozga.[[1](#_Literatura)][[3](#_Zaključak)]

Strojno učenje je važan dio današnjeg poslovnog i istraživačkog svijeta. Veliki napredak hardvera i softvera u posljednja dva desetljeća je omogućio širu upotrebu strojnog učenja. Zbog pada cijene uređaja za snimanje podataka (kao što su fotoaparat, mikrofon i drugi) bio je moguć razvoj algoritma koji na temelju velike količine podataka donosi pretpostavke. To je otvorilo cijelu sferu problema koji su prije bili prezahtjevni pa čak i nemogući za riješiti klasičnim programiranjem. U današnje doba većina ljudi se sreće s algoritmima koji su razvijeni uz pomoć strojnog učenja. Algoritmi koji se koriste u današnje doba imaju implementirano strojno učenje i prikupljanje podataka. Neke od najpoznatijih tvrtki koje tako funkcioniraju su Facebook i Google. Ova dva IT diva su izgradili svoje bogatstvo na prikupljanju informacija svojih korisnika. Oni koriste te podatke kako bi njihov algoritam korisnike svrstao u određene kategorije uz pomoć čega prilagođavaju korisnikovo iskustvo. U današnje doba informacije su jedna od najunosnijih sirovina, ali ne bi bile da nemamo automatiziran način za obradu i korištenje tih istih informacija.

Catherine Havasi, CEO tvrtke Luminoso, je u jednom intervjuu rekla da je cilj pomoći računalima i ljudima da se bolje međusobno razume. U današnje doba već koristimo strojno učenje u komercijalne svrhe baš iz tog razloga. [[4](#_Literatura)]

1. TENSORFLOW

To je platforma otvorenog koda koja omogućava jednostavan razvoj modela strojnog učenja. Nudi nekoliko razina apstrakcije što omogućava čak i početnicima korištenje. Razvio ga je Google i objavio 9. studenoga 2015. pod *Apache Licencom 2.0*. Od tada su uveli brojne promjene. Veliki razlog njegovog uspjeha jest u jednostavnosti korištenja. Razvili su cijelu zajednicu koja dijeli svoje resurse i znanje. Google je također razvio veliki broj svojih resursa za učenje i razvoj modela strojnog učenja kao što su brojni skupovi podataka i API. [[5](#_Literatura)]

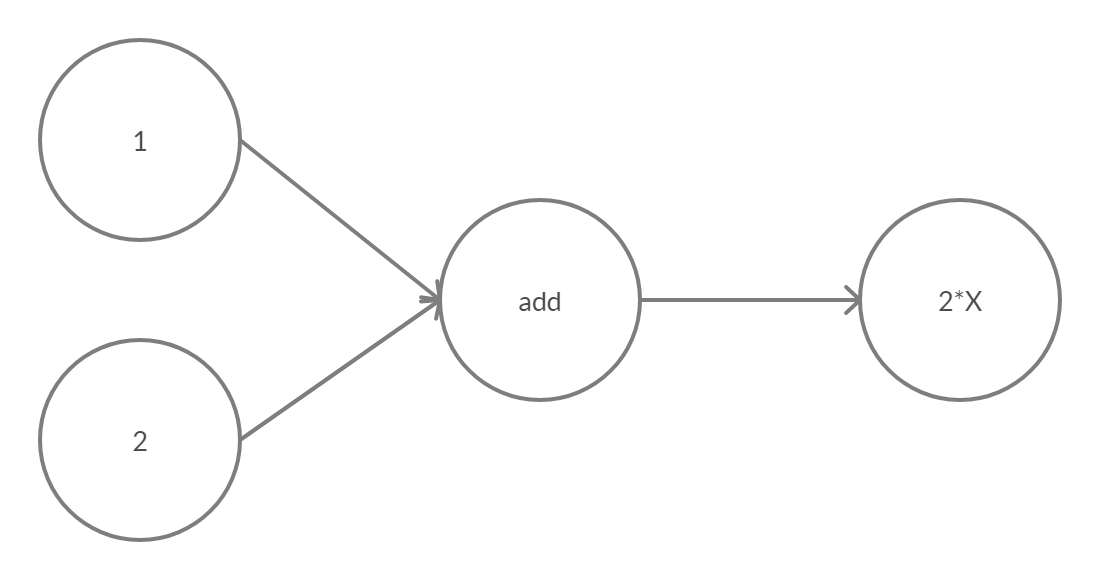
* 1. Podaci i operacije

Svi podaci se nalaze u objektima zvanim tenzori*.* Moguće je koristiti i druge strukture podataka kao što je matrica ili niz, ali ih treba prebaciti u tenzore kako bi ih mogli koristiti neuroni mreže. To je višedimenzionalni niz s više definiranih operacija nego običan niz. Svaki tenzor je definiran s jedinstvenim imenom, dimenzijom i vrstom podatka.

Dimenzije opisujemo uz pomoć ranga, ako imamo rang 1 tada je to jednodimenzionalni niz, ako je rang 2 onda je matrica itd. Vrsta podataka definira koji tip podataka se sprema. Npr. Tenzorranga 2 i tipa podataka float32 je dvodimenzionalna matrica koja pohranjuje decimalne brojeve veličine 32 bita.

Operacije koje se koriste su prilagođene za tenzore tako da se utječe na sve njihove članove. Npr. tf.add(x,y) zbraja sve članove za „x“ i „y“ koji imaju istu poziciju. „tf.“ označava biblioteku u kojoj je definirana operacija „add()“, a „x“ i „y“ su proizvoljni tenzori.[[11](#_Literatura)]

* 1. Kod kao graf

Možemo kod predstaviti kao graf koji se sastoji od čvorova i strelica koje ih povezuju kao što je prikazano na Slika 1. Čvorovi predstavljaju pojedine operacije kao zbrajanje, oduzimanje i brojne druge. Strelice označavaju kako podaci putuju među operacijama. [[11](#_Literatura)]

Slika 1 grafički prikaz koda

Ovo se naziva graf toka podataka*.*

* 1. Vrste

Postoji više vrsta TensorFlowa:

* TensorFlow
* TensorFlow.js – podržava programiranje u JavaScript.
* TensorFlow Lite – omogućava korištenje sučelja na mobilnim uređajima.
* TensorFlow Extended – služi za razvoj i upravljanje cjevovodom za strojno učenje.[[7](#_Literatura)]
  1. Jupyter notebook

Pruža internetsku aplikaciju prikladnu za bilježenje cijelog procesa:

* računanja
* razvoja
* dokumentiranja
* izvršavanje koda
* priopćavanje rezultata

Jupyterova bilježnica kombinira dvije komponente.

Web aplikacija: alat temeljen na pregledniku za interaktivno upravljanje dokumentima.

Dokumenti za bilježnice: prikaz cjelokupnog sadržaja vidljivog u web aplikaciji, uključujući ulaze i izlaze računanja, objašnjenja, matematiku, slike i prikaze objekata na bogatim medijima.[[11](#_Literatura)]

1. Google Colaboratory

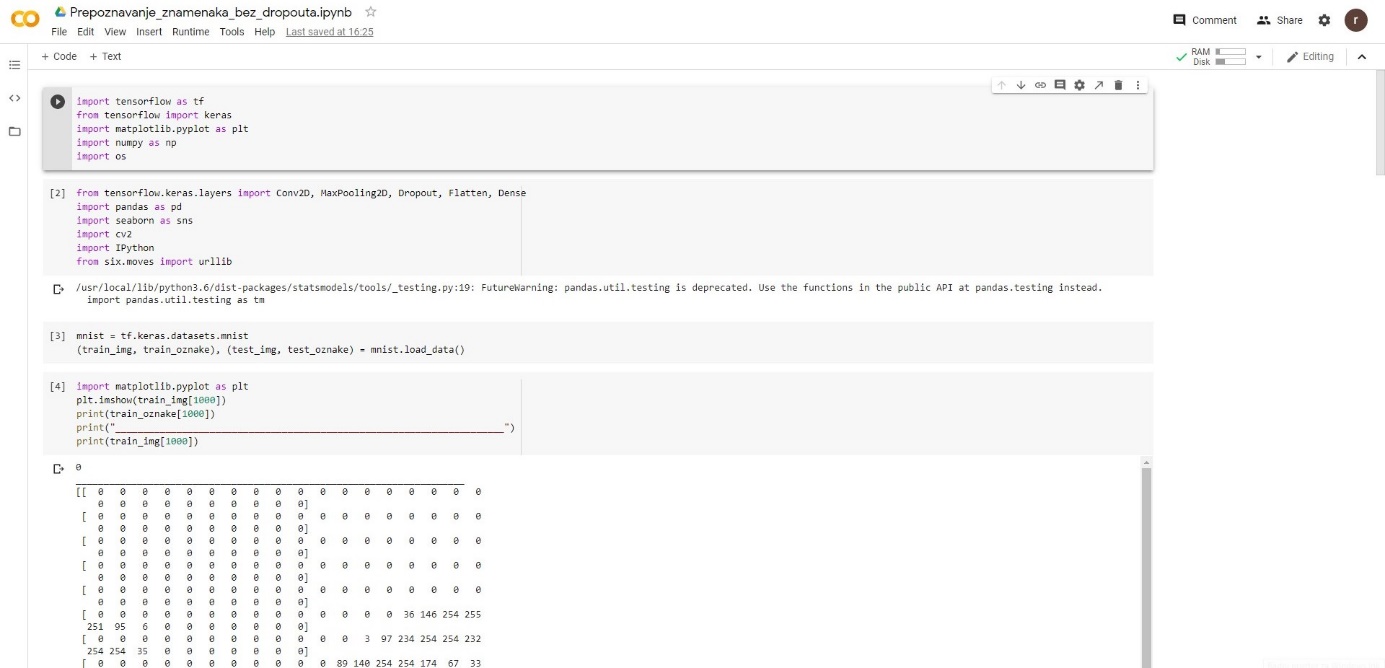
Colaboratory je Google-ovo besplatno okruženje za rad s Jupyter bilježnicama. Omogućava pisanje i pokretanje Pythona u Internet pregledniku na svom Google Disku. Nudi besplatan pristup grafičkoj kartici. Na taj način omogućava svima pristup tehnologiji za izradu i pokretanje procesorski zahtjevnih programa. Jako popularna funkcija je dijeljenje koda. Na jednostavan način se preko Google Diskračuna može slati kod, a također i prebaciti na GitHub.[[8](#_Literatura)]

* 1. Organizacija koda

Colaboratory omogućuje pisanje koda u jednostavne blokove baš poput Jupyter bilježnice. Na Slika 2 se vidi primjer podjele koda u blokove. S blokovima je moguće dijelove koda izvršavati neovisno o ostatku koda. Svaki blok može imati svoj ispis podataka što olakšava pronalazak grešaka jer možemo pratit rad svakog pojedinačnog bloka. Zbog ovakve organizacije kod je modularan što znači da se sastoji od lako zamjenjivih dijelova. Google je iskoristio tu modularnost te napravio bazu osnovnih funkcija koje možete jednim klikom ubaciti u svoju bilježnicu.

* 1. Hardver

Bilježnica je posebno dizajnirana za vrlo brzu i jednostavnu izradu prototipova. To su uspjeli tako što su omogućili pristup vrlo moćnom GPU-u i ukupno 12 GB RAM-a potpuno besplatno. Zbog toga nije bitna procesorska snaga sustava kojeg koristite, jer sva snaga dolazi sa strane Google-ovih servera.



Slika 2 Colaboratory

1. Strojno učenje

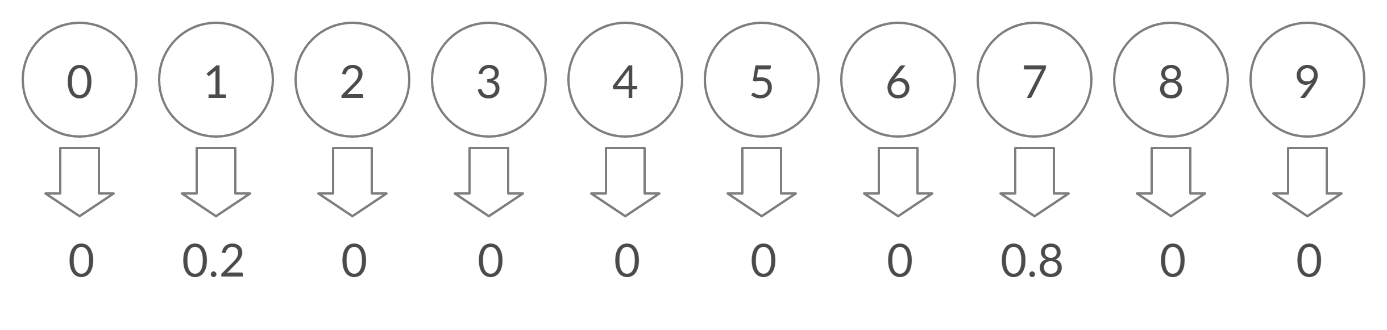
Strojno učenje označava da se problem ne rješava klasičnim programiranjem specifičnih zadataka nego učenjem i prilagođavanjem promjenjivim situacijama. Ovo je moguće uz pomoć različitih algoritama koji su specijalizirani za unaprjeđivanje u opisivanju i predviđanju rezultata.

* 1. Skup podataka

To je veliki skup uređenih podataka koji se koriste za strojno učenje i testiranje. U ovom primjeru se koristi skup podataka koji se sastoji od slika rukopisom napisanih znamenaka. Rezolucija slika je smanjena na 28×28 *piksela,* a boja promijenjena u sive tonove. To se napravi kako bi se smanjila količina podataka za obradu, a zadržale sve važne informacije.[[11](#_Literatura)]

* 1. Neuronske mreže

Ljudski mozak se sastoji od mreže neurona koji pri obavljanju nekog zadatka međusobno komuniciraju. Isti princip rada se koristi u neuronskim mrežama. Mreža se sastoji od umjetnih neurona. Oni su organizirani u više slojeva koji imaju različite funkcije. Mreža treba imati minimalno 3 sloja kako bi mogla funkcionirati: sloj za unos, skriveni sloj i sloj za ispis rezultata. Sloj za unos prima podatke, a za ispis ispisuje rješenje iz mreže. Skriveni sloj je bilo koji sloj koji nije sloj za unos ili ispis. On obrađuje podatke na temelju težine (*weight*) primijenjene na neuron. Veći broj skrivenih slojeva ne znači nužno da je mreža bolja jer s rastom broja slojeva opada brzina izvođenja. Za većinu jednostavnih problema dovoljan je jedan skriveni sloj. Broj neurona u pojedinom sloju ovisi o podacima koji se čitaju. U slučaju da sloj za unos učitava slike 28×28 *piksela* pa taj sloj ima barem 28 neurona. Sloj za ispis ovisi o tipu podatka koji ispisuje. Za kategorizaciju tj. za određivanje u koju kategoriju spada učitani podatak imat ćemo 1 neuron za svaku kategoriju. Za slučaj prepoznavanja znamenaka sloj ima 10 neurona jer ima 10 znamenaka (kategorija). Svaki neuron predstavlja jednu kategoriju i daje vjerojatnost da podatak spada u njegovu kategoriju. Zbroj vjerojatnosti svih neurona je 1. Npr. ako podatak ima 80% vjerojatnost da je 7 i 20% da je 1 onda je vjerojatnost ostalih kategorija 0 kao što je prikazano Slika 3.



Slika 3 Vjerojatnosti svih kategorija

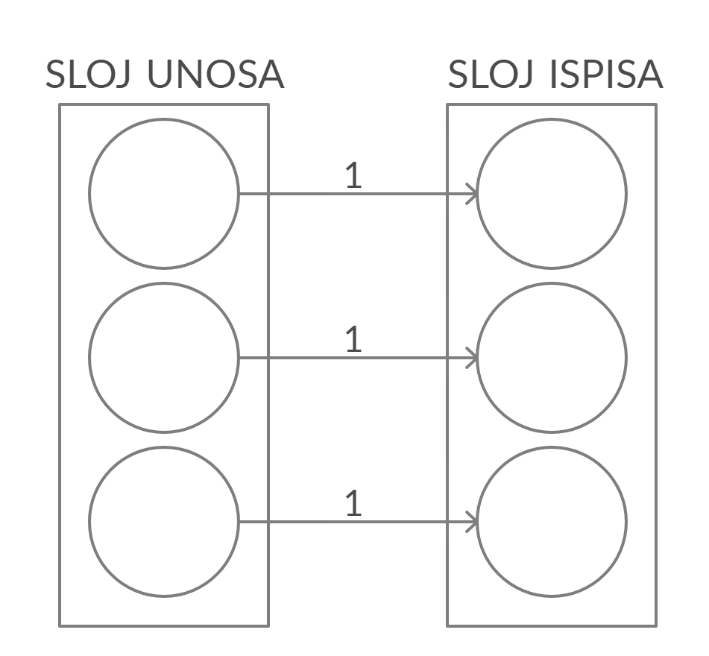
Ovaj model se zove umjetna neuronska mreža(ANN). Za vrijeme učenja veze među neuronima se formiraju i brišu te mreža uči koje su karakteristike važne, a koje ne.[[10](#_Literatura)][[11](#_Literatura)]

* 1. Konvolucijske neuronske mreže(CNN)

Jedne su od najčešće korištenih neuronskih mreža za učenje i analizu uz pomoć slika. CNN smanji rezoluciju slika i izradi mapu karakteristika. Mapa se izrađuje kako detektor karakteristika skenira i pronalazi karakteristike u slikama. Zbog ovoga slika gubi na rezoluciji jer broj karakteristika u mapi je manji od ukupnog broja na slici. Ovo ne predstavlja problem jer zadržava bitne karakteristike, a odbacuje nebitne.[[10](#_Literatura)][[11](#_Literatura)]

* 1. Težine neurona

Označavaju koje su veze među neuronima važnije od drugih. Na Slika 4 se nalazi mreža koja samo prosljeđuje podatke sa sloja za unos na sloj za ispis. Ima samo ta dva sloja i jednak broj neurona u oba. Povezani su neuroni na istim pozicijama kako bi dobili isti podatak na izlazu. Sve težine su 1.



Slika 4 Prijenos ulaza na izlaz

Proces kojim namještamo težine neurona se naziva propagacija pogreške unatrag.

* 1. Propagacija

Proces prolaska podataka kroz mrežu, ispisa pretpostavke te usporedba pretpostavke sa oznakom za trening se naziva propagacija prema naprijed.

Propagacija pogreške unatrag je dio treniranja u kojem se namještaju težine. Podaci putuju kroz mrežu unatrag te se izvodi pogreška za svaku težinu. Pogreška nam govori u kojem smjeru i koliko trebamo pomaknuti težinu te ih pomičemo.[[10](#_Literatura)]

* 1. Model mreže

Označava strukturu mreže tj. od kojih slojeva se sastoji.

model = tf.keras.models.Sequential(

[tf.keras.layers.Flatten(),

       tf.keras.layers.Dense(128, activation = tf.nn.relu),

       tf.keras.layers.Dropout(0.5),

       tf.keras.layers.Dense(128, activation = tf.nn.relu),

       tf.keras.layers.Dense(10, activation = tf.nn.softmax)])

Ovo je isječak iz koda.

Sequential označava sekvencijalni model mreže. *Flatten* označava sloj izravnavanja.

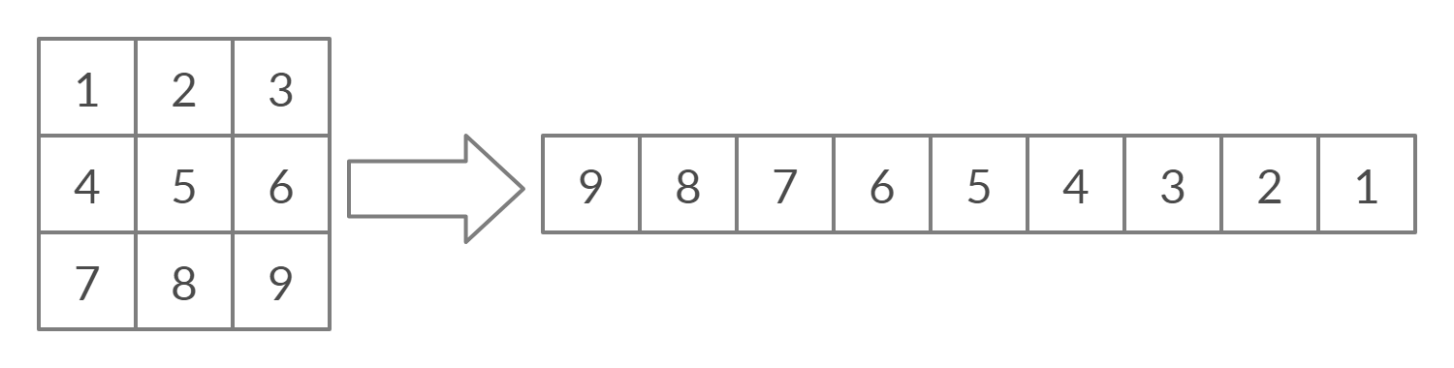
*Dropout* - funkciju isključivanja nasumično odabire neurone koji će se isključiti tu iteraciju. Služi kako bi se smanjila vjerojatnost efekta pretreniranja.

* 1. Sekvencijalni model

Sekvencijalni model mreže ima slojeve kroz koje podaci teku iz početnog u sljedeći i tako dalje do izlaznog sloja. Dobar je za klasifikaciju slika.

* 1. Redukcija dimenzije

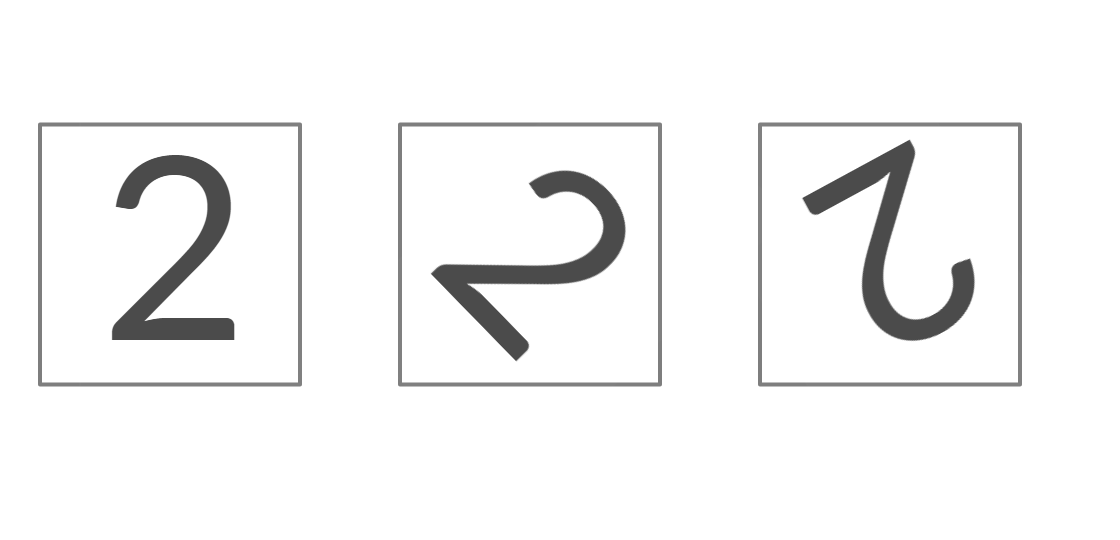
Ovaj sloj uzima dvodimenzionalni format podatka kao što je na Slika 5 i pretvara ga u jednodimenzionalni niz. Slika veličine 28×28 se pretvara u niz od 28\*28=784 *piksela.* Radimo ovo jer je mreži lakše raditi s tim formatom podataka.



Slika 5 Izravnavanje

* 1. Sažimanje

Sažimanje (*pooling)* pomaže u detekciji objekata na slici neovisno o rezoluciji slike, svijetlosti i drugim čimbenicima koji mogu utjecat na kvalitetu slike. Mreža se uči kako prepoznati uzorke neovisno o promjenjivim uvjetima. Rezultat ovog procesa se zove prostorna varijanca. Na Slika 6 se vidi primjer promjenjivih uvjeta. [[10](#_Literatura)]



Slika 6 Primjer promjenjivih uvjeta

* 1. Algoritam gradijentnog spusta

Upravlja optimizacijom težine pojedinih neurona u modelu mreže. Tijekom treniranja mreže postepeno sa svakom iteracijom smanjuje funkciju cijene. Funkcija cijene nam govori kolika je udaljenost pretpostavke od točnog rješenja, drugim riječima, što je cijena niža pretpostavka je točnija.[[10](#_Literatura)]

* 1. Prevođenje modela na strojni jezik

Prije treniranja mreže trebamo namjestiti proces učenja.

model.compile(optimizer = tf.keras.optimizers.Adam(),

              loss = 'sparse\_categorical\_crossentropy',

              metrics = ['accuracy'])

Ovo je isječak iz koda.

*Optimizer* označava koji algoritam koristimo za optimizaciju težina neurona za vrijeme propagacije unatrag. Uzima i prošle vrijednosti u obzir prilikom rada.

*Loss* funkcija – funkcija gubitka je suma svih pogrešaka koje model napravi za vrijeme učenja. Vrijednost 0 bi značila da je model savršen, ali u praksi se to ne dešava i može bit indikator efekta pretreniranja. Korištena je „sparse categorical crossentropy“ funkcija koja pretpostavlja da svaka slika ima točno jednu kategoriju kojoj pripada.

*Metrics* – označava koji parametar želimo pratiti. Vrijednost se ne prati za vrijeme samog učenja već nakon učenja.

* 1. Podešavanje modela

Prije učenja treba podesiti model.

model.fit(train\_img, train\_oznake, epochs = 20, validation\_split = 0.1)

*Train\_img* i *train\_oznake* označavaju slike za učenje i njihove pripadajuće oznake.

*Epoch* – označava broj iteracija učenja tj. koliko puta će model proći kroz podatke. Postavljen je na 20 što je relativno mali broj iteracija. Kompleksniji problemi i modeli koriste stotine pa čak i tisuće iteracija.

*Validation split* – je postotak koji označuje koliki dio slika za učenje će se izdvojiti za testiranje točnosti i funkcije gubitka na kraju svake iteracije.

1. Optimizacija

To je proces poboljšanja produktivnosti modela tijekom učenja. Postoji dosta načina optimizacije. Neki od njih su:

* funkcija isključivanja – *dropout*
* algoritam optimizacije Adam
* algoritam optimizacije Adamax
* algoritam optimizacije SGD
* algoritam optimizacije RMSprop

Ova četiri algoritma optimizacije su testirana nad istim modelom mreže. Prvo su testirani bez funkcije isključivanja, a zatim s njom. Njena funkcija je spriječiti efekt pretreniranja te poboljšati otpornost modela. Učenje je napravljeno nad 20 iteracija. Pri kraju svake iteracije je provjerena i spremljena funkcija gubitka i točnost uz pomoć malog postotka slika iz skupa za učenje. Tako dobivene vrijednosti se nazivaju:

* „train accuracy“ – točnost učenja
* „validation accuracy“ – potvrđena točnost
* „train loss“ – funkcija gubitka za vrijeme učenja
* „validation loss“ – potvrđena funkcija gubitka

Ovi algoritmi se zasnivaju na gradijentnom spustu – *gradient descent* (GD). Postoje 3 verzije gradijentnog spusta:

* BGD –Batch Gradient Descent
* SGD –Stochastic Gradient Descent
* Mini-batch Gradient Descent

BGP koristi cijeli skup podataka za učenje te na kraju cijele iteracije popravlja težine neurona.

SGD koristi jedan po jedan podatak (u ovom slučaju slika) te između učenja nad pojedinom podatku popravlja težine neurona.

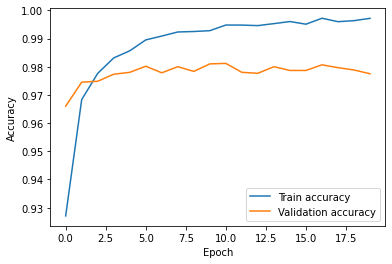
Mini-batch Gradient Descent je hibrid prethodna dva te koristi manje dijelove skupa podataka te nakon obrade tih podataka popravlja težine neurona te ide na sljedeći dio skupa.

Svi algoritmi optimizacije koji su ovdje obrađeni spadaju u SGD.[[9](#_Literatura)]

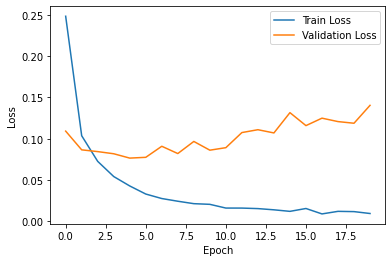
* 1. Adam

Adam je prilagođavajući algoritam za optimizaciju stope učenja. Dizajniran je za treniranje dubokih neuronskih mreža. On je kombinacija RMSprop i SGD-a. Računa zasebne stope učenja za različite parametre.

Prvo slijede podaci bez, a zatim s korištenjem funkcije isključivanja.

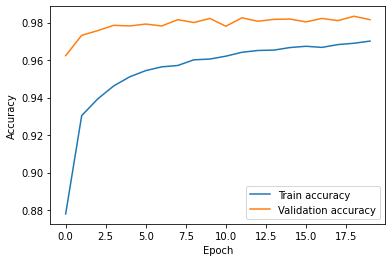


Slika 7 Adam bez funkcije isključivanja - točnost

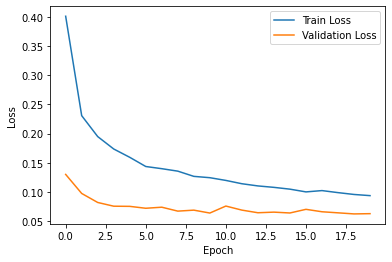


Slika 8 Adam bez funkcije isključivanja - funkcija gubitka

Ovaj algoritam ima veliku točnost kao što je vidljivo u Slika 7 i nisku funkciju gubitka na Slika 8. Niža vrijednost potvrđene točnosti i viša vrijednost potvrđene funkcije gubitka ukazuje na blagi efekt pretreniranja.



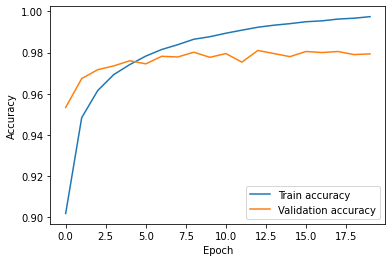
Slika 9 Adam s funkcijom isključivanja - točnost



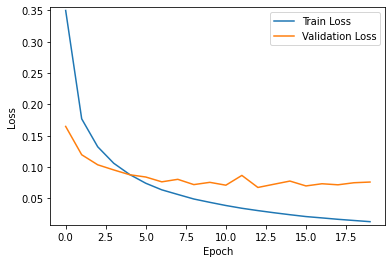
Slika 10 Adam s funkcijom isključivanja - funkcija gubitka

Dodavanjem funkcije isključivanja smo popravili efekt pretreniranja kao što se vidi iz grafova na Slika 9 i Slika 10.

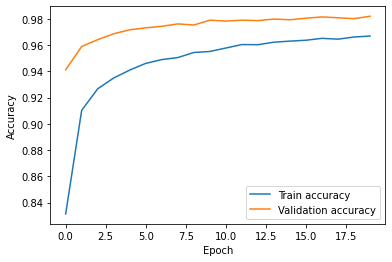
* 1. Adamax



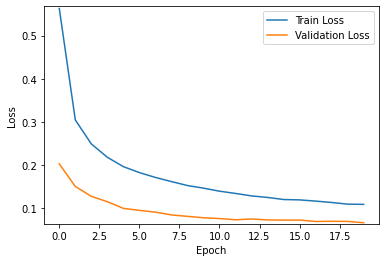
Slika 11 Adamax bez funkcije isključivanja - točnost



Slika 12 Adamax bez funkcije isključivanja - funkcija gubitka



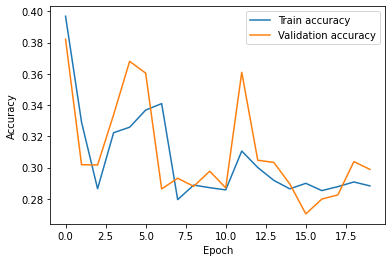
Slika 13 Adamax s funkcijom isključivanja - točnost



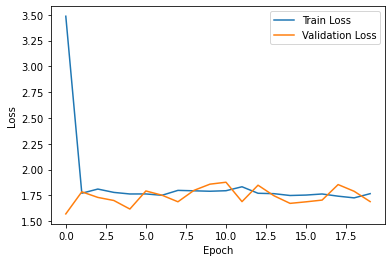
Slika 14 Adamax s funkcijom isključivanja - funkcija gubitka

Kao što vidimo učenje s algoritmom Adam i Adamax je gotovo jednako te su imali isti problem s efektom pretreniranja. To vidimo usporedbom Slika 11 i Slika 7 te Slika 12 i Slika 8.

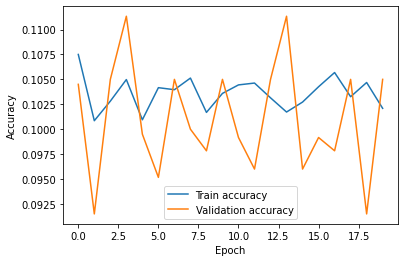
* 1. RMSprop



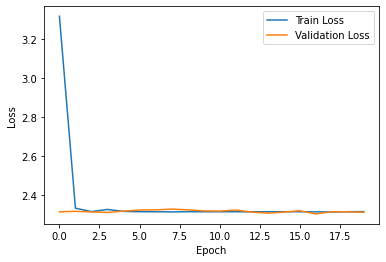
Slika 15 RMSprop bez funkcije isključivanja - točnost



Slika 16 RMSprop bez funkcije isključivanja - funkcija gubitka



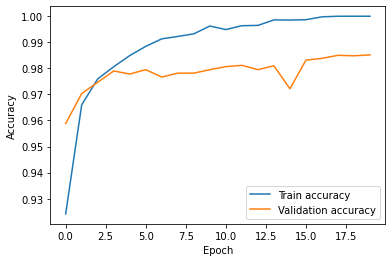
Slika 17 RMSprop s funkcijom isključivanja - točnost



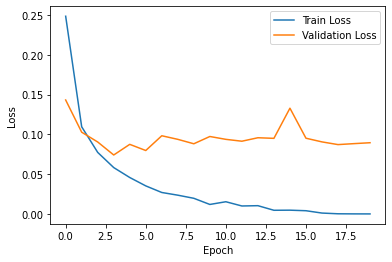
Slika 18 RMSprop s funkcijom isključivanja - funkcija gubitka

Nakon dodavanja funkcije isključivanja točnost se smanjila. To vidimo usporedbom Slika 15 i Slika 17.

* 1. SGD

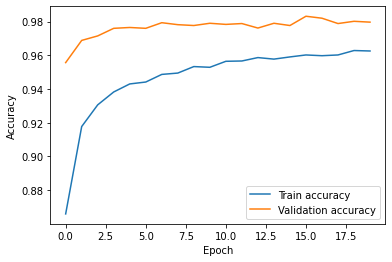


Slika 19 SGD bez funkcije isključivanja - točnost

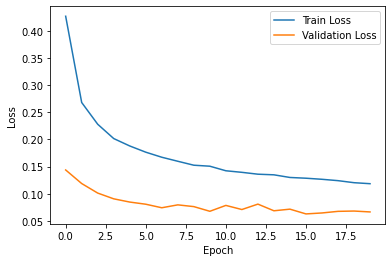


Slika 20 SGD bez funkcije isključivanja - funkcija gubitka

Iz grafa točnosti na Slika 19 se može očitati blagi efekt podnaučenosti. Uvođenjem funkcije isključivanja se to popravlja na Slika 21.



Slika 21 SGD s funkcijom isključivanja - točnost



Slika 22 SGD s funkcijom isključivanja - funkcija gubitka

Zaključak

Za usporedbu više algoritama optimizacije potrebno je prvo napraviti funkcionalan model nad kojim se može vršiti testiranje. U ovom slučaju smo imali dva modela koji se razlikuju po jednom pozivu funkcije ispadanja. Na taj način smo mogli vidjeti kako se algoritmi ponašaju u različitim okruženjima.

Na temelju svih prikupljenih podataka jedini algoritam optimizacije koji se izdvojio je RMSprop. Imao je najnižu točnost s jako visokom vrijednosti funkcije gubitka.

Ne može se na temelju prikupljenih podataka vidjeti koji je algoritam najbolji. Tri preostala algoritma imaju jako slične rezultate i vrijeme izvršavanja im je također jednako. Adam i Adamax su se pokazali jednostavnijim za korištenje. Za daljnje testiranje treba uzeti kompleksniji problem kako bi vidjeli razlike između algoritama :Adam, Adamax i SGD.

Literatura

[1] Uvod dio s povjesti i godinama: <https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/#2682e78115e7>

[2] Arthur Lee Samuel: <https://history.computer.org/pioneers/samuel.html>

[3] Frank Rosenblatt: <https://dl.acm.org/doi/abs/10.5555/1074100.1074686>

[4] Uvod dio s Catherine Havasi <https://www.wired.com/insights/2014/09/artificial-intelligence-algorithms-2/>

TensorFlow:[5] <https://www.tensorflow.org/about>

[6] <https://www.tensorflow.org/about/case-studies>

[7] <https://www.tensorflow.org/learn>

[8] Colaboratory: <https://colab.research.google.com/notebooks/intro.ipynb>

[9] Algoritmi optimizacije: <https://www.researchgate.net/profile/Eustace_Dogo/publication/330760888_A_Comparative_Analysis_of_Gradient_Descent-Based_Optimization_Algorithms_on_Convolutional_Neural_Networks/links/5d033d5aa6fdccd13099362d/A-Comparative-Analysis-of-Gradient-Descent-Based-Optimization-Algorithms-on-Convolutional-Neural-Networks.pdf?origin=publication_detail>

[10] Mikko Makela – Understanding, Building and Analyzing Basic Convolutonal Neural Networks – 2019 godine izdano

[11] Nishant Shukla – Machine learning with TensorFlow – verzija 10

Popis oznaka i kratica

API – aplikacijsko programsko sučelje

BGD –Batch Gradient Descent – vrsta algoritma optimizacije

CNN – konvolucijska neuronska mreža

Epoch – iteracija

GPU – grafička kartica

Luminoso - tvrtka koja se bavi analizom teksta i umjetnom inteligencijom.

Mini-batch Gradient Descent - vrsta algoritma optimizacije

Node – neuron

RAM – radna memorija

SGD –Stochastic Gradient Descent - vrsta algoritma optimizacije

Train\_img – ime datoteke koja sadrži slike za treniranje

Train\_oznake – ime datoteke koja sadrži oznake slika za treniranje

Weight – težina neurona

Sažetak

U ovom završnom radu prikazane su osnovne značajke i koncepti strojnog učenja kroz analizu: (i) konvolucijske neuronske mreže, te (ii) procesa učenja i testiranja. U uvodnom dijelu predstavljeno je okruženje TensorFlow, u kojem je na primjeru prepoznavanja znamenki MNIST slikovne baze demonstrirano razumijevanja procesa dubokog učenje. U završnom dijelu rada izloženi su rezultati učenja CNN mreže korištenjem postupka optimizacije sa i bez *dropout* funkcije i njihova usporedba.

Ključne riječi

Konvolucijska neuronska mreža, strojno učenje, algoritam optimizacije, TensorFlow

Summary

This bachelor thesis deals with the basic features and concepts of machine learning through the analysis of: (i) convolutional neural networks, and (ii) learning and testing processes. In the introductory part, the TensorFlow environment is presented, through which the understanding of the deep learning process is demonstrated on the example of recognizing the handwritten digits of the MNIST database. The results of learning CNN networks using the optimization procedure with and without a dropout function and their comparison are presented.

Keywords

Convolutional neural networks, machine learning, optimizer, TensorFlow

Title

The development and analysis of convolutional neural networks