SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6970

METODE OPTIMIZACIJE PARAMETARA KONVOLUCIJSKIH NEURONSKIH MREŽA

Marin Ovčariček

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6970

METODE OPTIMIZACIJE PARAMETARA KONVOLUCIJSKIH NEURONSKIH MREŽA

Marin Ovčariček

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

ZAVRŠNI ZADATAK br. 6970

Pristupnik:	**Marin Ovčariček (0036506412)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	doc. dr. sc. Marko Đurasević

Zadatak: **Metode optimizacije parametara konvolucijskih neuronskih mreža**

Opis zadatka:

Proučiti konvolucijske neuronske mreže te moguće metode koje se mogu primijeniti za optimizaciju njenih parametara. Odabrati i opisati najčešće korištene metode za optimizaciju parametara neuronskih mreža iz literature. Oblikovati i ostvariti programski sustav za učenje konvolucijskih neuronskih mreža odabranim postupcima optimizacije. Ocijeniti učinkovitost odabranih metoda optimizacije na problemu prepoznavanja rukom pisanih znamenki. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

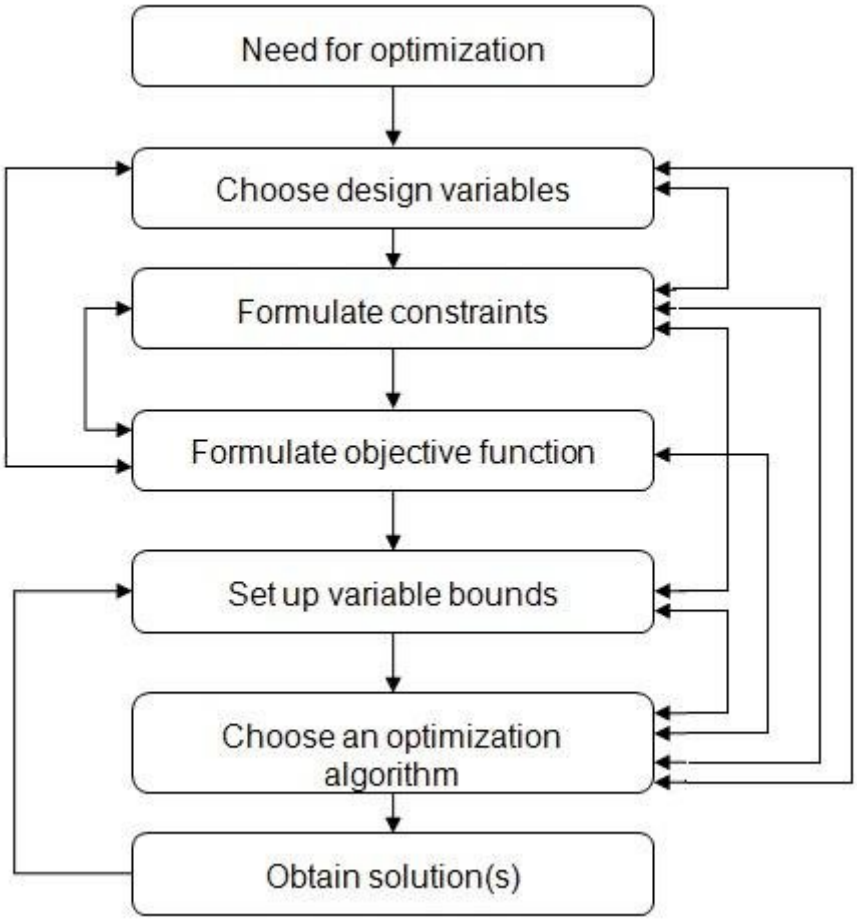
Rok za predaju rada: 12. lipnja 2020.

Zahvaljujem mentoru dr. sc. Marku Đuraseviću na pomoći i savjetima tijekom studiranja i izrade ovog rada

SADRŽAJ

1. Uvod

Optimizacijske metode su procedure koje se izvode iterativno i modificiraju rješenje dok se ne postigne optimum ili zadovoljavajući rezultat. Potreba za optimizacijom se javlja gdje god je potrebna neka vrsta efikasnosti. Kako bi se nešto optimiziralo prvo se trebaju postaviti varijable koje će se razmatrati za vrijeme optimizacije. Zatim se trebaju postaviti ograničenja na varijable (konačan skup vrijednosti) i na kraju izabrati optimizacijski algoritam koji će kroz iteracije dati rješenje. Slika 1.1. daje prikaz tijeka općenitog postupka optimizacije kao dijagrama sa stanjima i prijelazima. Umjetna neuronska mreža je model koji se često koristi u raznim sferama računarstva. No, kako bi mreža davala zadovoljavajuće rezultate za određeni problem, potrebno je optimirati njene parametre. U ovom radu razmatrat će se razne metode optimiranja bazirane na gradijentima kojima bi se mreža što bolje prilagodila problemu. U drugom poglavlju rada daje se kratko objašnjenje što su umjetne neuronske mreže, od čega se sastoje, kako uče i za rješavanje kakvih problema se koriste. Također dodatno su objašnjene specifičnosti konvolucijske neuronske mreže nad kojom će se raditi optimizacija. U trećem poglavlju postavlja se problem kojeg će mreža rješavati. To uključuje arhitekturu mreže, opis skupa podataka za učenje i testiranje, toka podataka te varijable koje će se optimizirati i njihova ograničenja. Objašnjena je ideja algoritma *backpropagation* te su ukratko objašnjene metode učenja uz popratne formule. U četvrtom poglavlju prezentirani su rezultati optimizacije kroz tablicu izabranih kombinacija varijabli, grafa točnosti kroz vrijeme za svaku metodu učenja te kutijasti dijagram s prikazom točnosti svih metoda. U petom poglavlju su izneseni pregled rada i zaključak o postignutom rezultatu.



Slika 1.1: Općeniti optimizacijski postupak

2. Umjetne neuronske mreže

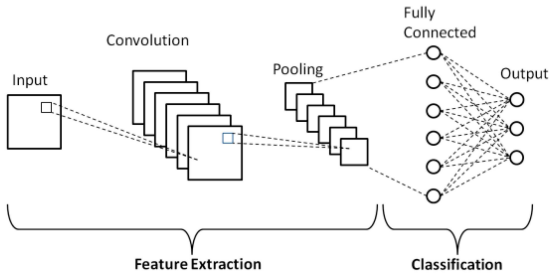
U današnje vrijeme sve je veća potreba za alatima koji raspoznaju uzorke. Uzorci mogu biti zvuk, slika, kretanje dionica, promjena vremena i slično. Umjetne neuronske mreže pokazale su se kao snažan alat za rješavanje problema raspoznavanja, predviđanja i klasificiranja. Razlog uspjeha neuronskih mreža je mogućnost koreliranja podataka koji su nelinearno povezani, spremanje sakupljenog znanja te korištenje tog istog znanja za suočavanje s neviđenim problemima.

Umjetna neuronska mreža (engl. *Artificial Neural Network*) je sustav za računanje inspiriran stvarnim neuronskim mrežama kod životinja. Umjetni neuroni isto kao i pravi imaju mogućnost primanja, obrađivanja te prosljeđivanja informacije u mreži te mogućnost povezivanja s proizvoljnim brojem drugih neurona. Prosljeđivanje informacija (engl. *feedforward*) odvija se tako da svaki neuron zbroji doprinose koje mu šalju drugi neuroni, obradi ih aktivacijskom funkcijom te proslijedi dalje. Mreža se sastoji od slojeva neurona koji mogu obavljati različite uloge. Najopćenitiji mreža je oblika: ulazni sloj koji prima podatke za kojim slijede proizvoljan broj sakrivenih slojeva koji mogu obavljati različite funkcije te izlazni sloj koji daje izlazne podatke (engl. *output*). Znanje je ostvareno jačinom povezanosti između neurona (engl. *weights*) koji su uz arhitekturu mreže i unesene podatke zaslužni za informaciju koju mreža daje. Učenje je mijenjanje jačina povezanosti neurona kako bi se mreža što bolje prilagodila ulaznim podatcima i davala točnije rezultate. Jedan od postupaka kojim se može obavljati učenje je *backpropagation*.

2.1. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže su potklasa dubokih neuronskih mreža. Najviše se koriste kod raspoznavanja slika, ali imaju primjene i kod raspoznavanja zvuka, analize medicinskih slika i obrade prirodnog jezika. Snaga konvolucijskih mreža dolazi iz translacijske invarijacije, odnosno mogućnosti raspoznavanja uzoraka bez obzira na njihov položaj u ulaznim podatcima, prepoznavanja složenih uzoraka uz pomoć kombiniranja

manjih uzoraka te neovisnosti o prijašnjem znanju te samostalnog učenja uzoraka što joj daje prednost nad ručno rađenim algoritmima. Konvolucijske neuronske mreže se obično sastoje od jednog ili više "blokova" slojeva za konvoluciju i sažimanje, potpuno povezanog te izlaznog sloja. Konvolucijski sloj prima podatke u obliku trodimenzionalne matrice gdje duljina i širina mogu biti veličina slika a dubina RGB kanali boja. Konvolucijski sloj prolazi po ulaznim podatcima svojim "vidnim poljem" (engl. *receptive field*) , uz pomoć filtara obavlja konvolucije te stvara mapu značajki (engl. *feature map*) i prosljeđuje ju pooling sloju. Pooling sloj svojim vidnim poljem obavlja pool operaciju kojom u manju mapu značajki izdvaja samo najznačajnije elemente. Neki od načina sažimanja su sažimanje maksimalnom, minimalnom i srednjom vrijednošću. Potpuno povezani sloj sakuplja sve mape značajki pooling sloja u jedan vektor te prosljeđuje informacije sljedećim potpuno povezanim slojevima ili izlaznom sloju. Ovdje se *feedforward* obavalja identično kao kod obične neuronske mreže uz pomoć težina između neurona i aktivacijskih funkcija. Izlazni sloj uz pomoć primljenih informacija daje rezultat. Na slici 2.1 prikazana je općnita arhitektura konvolucijske neuronske mreže.



Slika 2.1: Općenita konvolucija neuronska mreža

2.2. Backpropagation

Backpropagation je vrsta algoritama kojima se za vrijeme učenja mreže nastoji smanjiti ukupna greška u izlaznim podatcima. Kod *backpropagationa* se prvo provodi *forward pass*, odnosno informacija se šalje u mrežu, obrađuje se u slojevima i dobiva se rezultat na izlaznom sloju. Nakon toga se računa greška izlaza prema unaprijed zadanoj funkciji greške. Na temelju izračunate greške računa se gradijent. Gradijentu se pridodaje negativan predznak jer je cilj minimizirati funkciju greške te se postupkom *backward pass* propagira kroz slojeve pomoću različitih metoda učenja te pravila derivacija kompozicija funkcija i derivacija aktivacijskih funkcija. Pomoću izračunatih

grešaka svakog sloja mijenjaju se težine između neurona kako bi se mreža prilagodila informacijama.

3. Metode optimizacija

Za vrijeme optimizacije mogu se razmatrati parametri, hiperparametri i strategije. Parametri su varijable koje se inicijaliziraju na početku te ih model sam kroz učenje trenira. U ovom slučaju to su težine između neurona i vrijednosti filtara u konvolucijskom sloju. Hiperparamteri su varijable koje se zadaju na početku i ne mijenjaju se za vrijeme učenja. Neki od mogućih hiperparametara su:

- Broj slojeva - jedan od glavnih problema odabira hiperparametara, manji broj slojeva će dati svojstva bolje generalizacije i učiniti mrežu bržom, ali premali broj može uzrokovati loše klasificiranje ulaznih podataka
- Broj filtara u konvolucijskom sloju
- Stopa učenja - određuje koliko će značajne biti promijene težina neurona za vrijeme *backpropagationa*, prevelika stopa učenja će uzrokovati oscilacije u točnosti (divergenciju) i nemogućnost pronalaska zadovoljavajućeg lokalnog minimuma funkcije greške, dok će premala stopa značajno usporiti konvergenciju prema minimumu
- Konstante za računanje momentuma
- Aktivacijske funkcije - razne aktivacijske funkcije daju različita svojstva mreži, za odabir aktivacijskih funkcija treba razmisliti o njihovom redoslijedu te efikasnosti i brzini učenja.
- Funkcija greške
- Veličina serije (engl. *Batch size*) - serija ja je niz podataka nad kojim se provodi učenje prije nego što se akumulirane promjene težina ažuriraju, dobra praksa kod učenja kada je ukupan broj ulaznih podataka jako velik je rasporediti podatke u manje serije, preveliki *batch size* može uzrokovati pregeneraliziranost, odnosno mreža se neće moći dobro prilagoditi novim podatcima
- Broj epoha - označava broj koliko puta će mreža učiti nad cijelim skupom po-

dataka, broj epoha se može postaviti da bude veliki i onda staviti uvjet da se prekine učenje kod određenog postotka točnosti ili pokretati mrežu više puta povećavajući broj epoha dok se ne postigne željena točnost.

- *Dropout vjerojatnost* - inspiriran stvarnim svijetom gdje kroz vrijeme ljudi teže uče nova znanja, dropout predstavlja gašenje određenih neurona za pojedini korak kako bi mreža bolje generalizirala

Strategije koje se mogu koristiti uključuju razne načine inicijaliziranja parametara, normalizacija ulaznih podataka na neki interval, standardizacija ulaznih podataka (rasporediti ih u normalnu distribuciju), te metode učenja.

3.1. Varijable i ograničenja

U ovom radu arhitektura mreže, način inicijalizacije parametara, vrijednosti za računanje momentuma, aktivacijska i funkcija greške su iste za svaku mrežu. Dropout radi jednostavnosti implementacije nije korišten. Kombinacija parametara koja će se pokušati optimizirati kako bi mreža davala što točniji rezultat su: *batch size* iz skupa vrijednosti {1, 50 100, 500 i 1000}, broj epoha {5,10}, stopa učenja (na dekadskoj logaritamskoj skali u intervalu <-5,-3>) te metoda učenja. Metode učenja korištene u ovom radu su: Gradijentni spust, Momentum, ADAM, Nadam, RMSProp, AdaGrad.

3.2. Matematičke oznake

Oznake koje će se koristiti u formulama za metode učenja su:

- $\nabla J(\theta_t)$ ili g_t -gradijent funkcije greške
- η - stopa učenja (eta)
- θ - težine između neurona (theta)
- ϑ, m - iznos momentuma (malo slovo theta,m)
- β, γ - konstante za računanje momentuma (beta, gamma)
- ϵ - sklonost (eng. *bias*, epsilon)

Formule za računanje promjena težina neurona su preuzete sa internetskih članaka: (Hansen) i (Ruder).

3.3. Gradijentni spust

Gradijentni spust je optimizacijski algoritam nalaženja minumuma funkcije. Bazira se na spuštanju po najvećoj strmini.

$$\theta_{t+1} = \theta - \eta \nabla J(\theta_t)$$

3.4. Momentum

Momentum je nadogradnja na gradijentni spust gdje se na ukupnu promjenu nadodaje izraz koji se računa uz pomoć konstante i doprinosa prijašnjeg koraka. Prednost momentuma je brža konvergencija dok je nedostatak mogućnost promašivanja lokalnog minimuma funkcije greške te osciliranje oko tog minimuma.

$$\theta_{t+1} = \theta - \eta \nabla J(\theta_t) + \gamma \vartheta_t$$

$$\vartheta_t = \eta \nabla J(\theta_t)$$

3.5. ADAM

Adaptivni momentum je nadogradnja na Momentum algoritam gdje je momentum umnoškom vezan za learning rate odnosno jačina promjene veza između neurona će ovisiti o trenutnom i o prošlim koracima.

$$\theta_{t+1} = \theta_t - \frac{\eta \cdot \hat{m}_t}{\sqrt{\hat{\vartheta}_t} + \epsilon}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{\vartheta}_t = \frac{\vartheta_t}{1 - \beta_2^t}$$

$$m_t = (1 - \beta_1) g_t + \beta_1 m_{t-1}$$

$$\vartheta_t = (1 - \beta_2) g_t^2 + \beta_2 \vartheta_{t-1}$$

3.6. AdaGrad

AdaGrad postupno smanjuje learning rate tako što ga dijeli s korijenom sume kvadrata prošlih koraka. Razlog tome je da u kasnijim stadijima učenja treba raditi manje korake

kako se ne bi promašio lokalni minimum.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\epsilon + \sum_{\tau=1}^{t}(\nabla J(\theta_{\tau,i}))^2}} \nabla J(\theta_{t,i})$$

3.7. RMSProp

Za razliku od AdaGrada, RMSProp omogućuje dizanje ili spuštanje learning ratea ovisno o ovom i prošlim koracima.

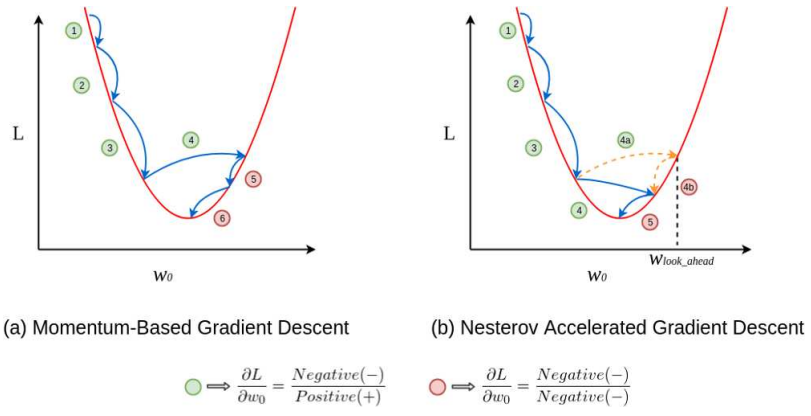
$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\epsilon + E[g^2]_t}}$$

$$E[g^2]_t = (1-\gamma)g^2 + \gamma E[g^2]_{t-1}$$

3.8. Nadam

Nadam (Nesterov-accelerated Adaptive Moment Estimation) kombinira ADAM i NAG (Nesterov accelerated gradient) kako bi predvidio kretanje funkcije greške i bolje odredio buduće promjene jačina povezanosti neurona. Na slici 3.1 se može vidjeti prednost Nesterov tehnike koja manjim koracima omogućuje brže približavanje lokalnom minimumu.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{\vartheta}_t} + \epsilon}(\beta_1 \hat{m}_t + \frac{(1-\beta_1)g_t}{1-\beta_1^t})$$



Slika 3.1: Usporedba gradijentnog spusta uz pomoć momentuma i Nesterov tehnike

4. Opis problema

Za problem optimizacije parametara konvolucijskih neuronskih mreža uzet je slučaj prepoznavanja ručno pisanih znamenki. Koristi se jednostavna konvolucijska mreža koja se sastoji od jednog konvolucijskog, pooling, potpuno povezanog i izlaznog sloja. Mreža se uči i testira nad skupom podataka (LeCun i Cortes, 2010). Skup se sastoji od skupa za učenje od 60 000 slika i skupa za testiranje od 10 000 slika, od kojih se 5 000 nalazi i u skupu za učenje. Svaka slika znamenke je veličine 28x28 piksela u *grayscale* formatu te je centrirana prema magnitudi piksela. Prije svake epoha skup ulaznih podataka se pseudonasumično izmiješa.

Prije slanja u mrežu, na svakoj slici se obavlja padding na veličinu 32x32 te se šalje u konvolucijski sloj. Konvolucijski sloj obrađuje slike uz pomoć 10 filtera veličine 5x5 čije su vrijednosti na početku pseudonasumično inicijalizirane na interval <-0.001, 0.001> te stvara mape značajki veličine 28x28. Pooling sloj obavlja operaciju maxpool nad mapama značajki s vidnim poljem 2x2 i pomakom 2 te daje izlaz veličine 14x14. Potpuno povezani sloj sakuplja sve izlaze iz pooling sloja u vektor od 1960 elemenata te predaje vrijednosti izlaznom sloju koji na temelju toga daje vektor predikcije znamenki s jednim elementom za svaku znamenku, sveukupno 10. Znamenkama se pridružuje neuron po principu - vrijednost znamenke je n i njoj je pridružen n-ti neuron u izlaznom sloju. Predikcija znamenki se računa kao *maxpool* vektora vjerojatnosti koji je dobiven kroz *Softmax* funkciju. Softmax funkcija prima kao argument vektor vrijednosti iz izlaznog sloja i normalizira ga u distribuciju vjerojatnosti proporcionalnu vrijednostima eksponecijalne funkcije gdje su argumenti vrijednosti iz vektora. Kao funkcija greške koristi se *Cross entropy* funkcija, te njen gradijent za unazadnu propagaciju.

4.1. Nalaženje optimalnih hiperparametara

Optimizacija mreže u ovom radu provodila se u dva koraka. U prvom koraku se ograničava skup vrijednosti parametara koji će se isprobavati kako bi se pronašla optimalna

kombinacija za svaku metodu učenja. Nadalje, kako bi se izbjegla "kombinatorna eksplozija" zbog broja hiperparametara i veličina skupa vrijednosti za optimizaciju, hiperparametri se optimiziraju jedan po jedan. Konkretno, prvo se uzimaju različite vrijednost jednog hiperparametra, vrijednosti ostalih parametara se drže konstantnima te se gleda prosječna točnost mreže nakon određenog broja pokretanja. Najbolja vrijednost se uzima kao konstanta te se prelazi na sljedeći hiperparametar po prioritetu, postupak se ponavlja dok se ne iscrpe svi hiperparametri. U ovom radu hiperparametri su prioritetnim redom *batch size*, broja epoha te stopa učenja. Za odabir svakog hiperparametra mreža se pokretala 5 puta.

4.2. Testiranje metoda učenja

Nakon što su za svaku metodu učenja izabrani hiperparametri u drugom koraku se najbolje dobivene kombinacije primjenjuju na mrežu. Mreža se pokreće 10 puta te se rezultati uzorkuju u vremenskim trenutcima i uprosječuju kako bi se prikazalo kretanje preciznosti mreže tijekom učenja. Pomoću dobivenih rezultata razmatrala se brzina učenja, monotonost rasta preciznosti te raspršenost točnosti za svaku metodu učenja.

Metoda učenja	*Batch size*	Broj epoha	Stopa učenja
Gradijentni spust	50	5	0.0001
Momentum	50	5	0.0001
ADAM	50	5	0.0001
Nadam	1	5	0.00005
AdaGrad	1	5	0.001
RMSProp	1000	10	0.001

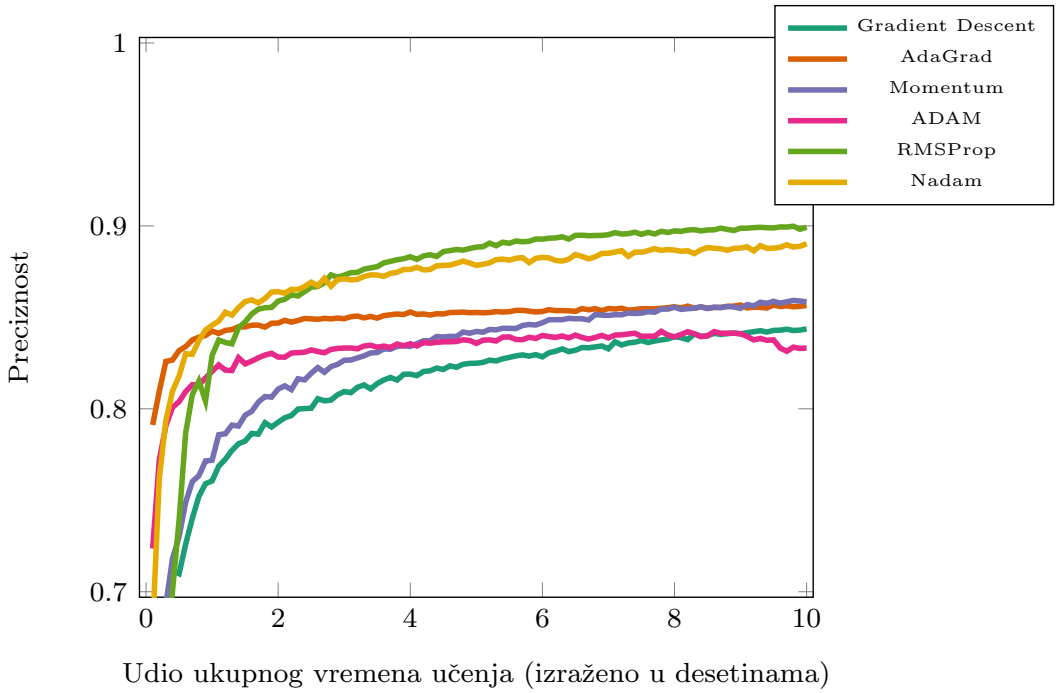
Tablica 4.1: Kombinacije metoda učenja i parametara mreže.

Rezultati drugog koraka optimizacije predstavljeni su kroz tablicu, graf točnosti kroz vrijeme te kutijasti dijagram.

	AdaGrad	ADAM	Gradient desc.	Momentum	Nadam	RMSProp
Min	0.8532	0.8271	0.8309	0.8217	0.8563	**0.8786**
Avg	0.8560	0.8328	0.8441	0.8589	0.8905	**0.8993**
Max	0.8584	0.8426	0.8766	0.8985	0.9174	**0.9182**

Tablica 4.2: Minimlana, prosječna i maksimalna točnost metoda učenja.

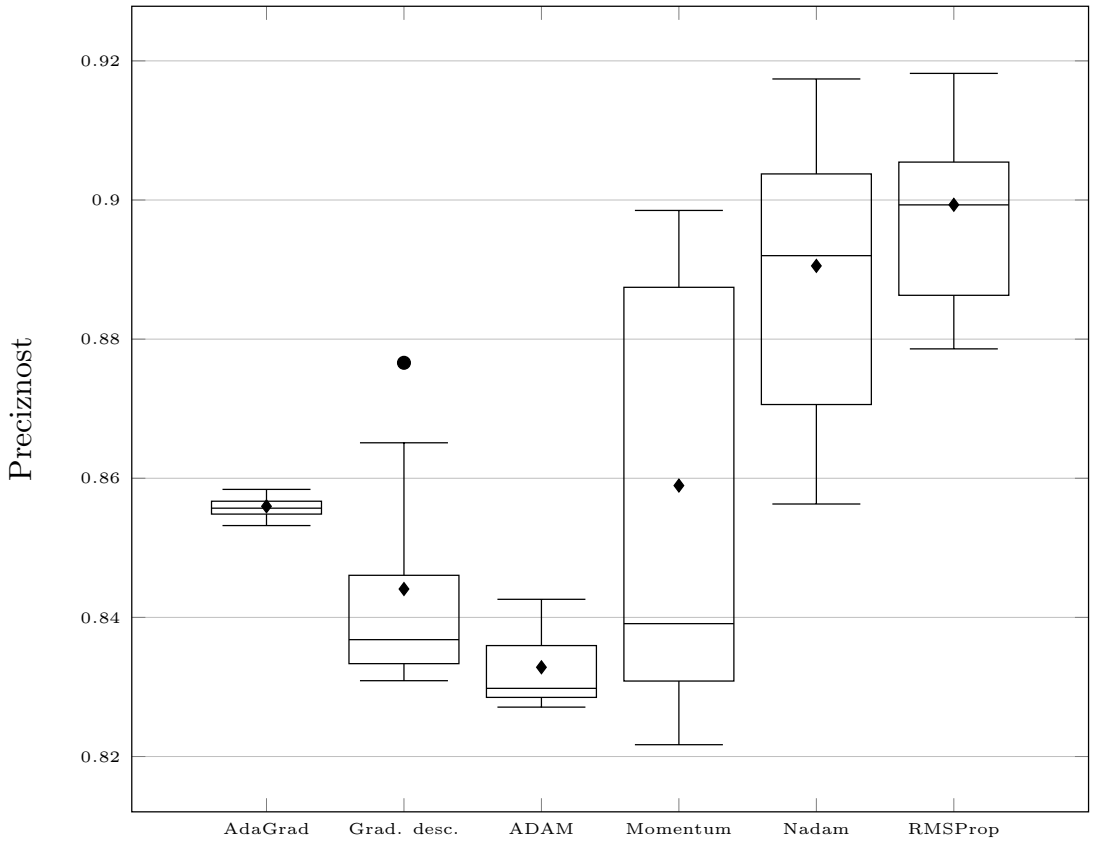
Iz tablice 4.1 se može vidjeti kako različite metode učenja dobre rezultate dobivaju za razne kombinacije hiperparametara koje su mogu razlikovati u više redova veličina. *Batch size* i stopa učenja su hiperparametri koji zahtijevaju promišljanje o tome kako će promijene u njihovim vrijednostima utjecati na rezultat. Tablica 4.2 prikazuje najmanje, najveće i prosječne točnosti za svaku metodu učenja iz čega je vidljivo da je metoda RMSProp postigla najbolje rezultate. Kao što se može vidjeti na slici 4.1 metode učenja postižu većinu svoje točnosti prije polovice cjelokupnog perioda učenja, bez obzira na to broj epoha je veći kako bi se provjerilo dugoročno ponašanje metoda učenja.



Slika 4.1: Kretanje točnosti kroz vrijeme za pojedine metode učenja

Kao što je i očekivano, metoda gradijentnog spusta imala je najsporiji rast točnosti. Nadogradnja metodom Momentum ubrzala je kretanje prema lokalnom minimumu funkcije greške te dala bolji rezultat. Metoda ADAM koja je nadogradnja na

metodu Momentum, značajno je u početku ubrzala rast točnosti mreže, dok je na kraju ipak završila s malo najslabijim rezultatom, stagnacija točnosti i pad pri kraju ukazuju na neoptimalno izabrane hiperparametre. Metoda AdaGrad je imala najbrži rast zbog visoke početne stope učenja, stabilnost u kasnijim koracima joj je omogućilo monotono smanjivanje stope učenja kako se ne bi krenulo u krivom smjeru, odnosno smanjivanju točnosti. Metoda Nadam postigla je drugu najveću točnost kombinirajući svojstva momentuma i korekcije uz pomoć *Nesterov Accelerated Gradient*. RMSProp metoda je na postigla najveću točnost iako bi u početku imala manje oscilacije. Dobar rezultat omogućilo joj je svojstvo dizanja i spuštanja stope učenja po potrebi, što je dovelo do preciznijeg ugađanja težina neurona. Sve metode su u početku imale nagli rast točnosti te su kasnije zbog svojih i mrežinih ograničenih sposobnosti učenja usporile rast, ali su zadržale monotoni rast ukupne točnosti.



Slika 4.2: Kutijasti graf točnosti metoda

Kutijasti dijagrami prikazani na slici 4.2 daju dodatne informacije o ponašanju određenih metoda učenja. AdaGrad metoda se pokazala kao najkonzistentnija metoda, odnosno imala je najmanji raspon točnosti na kraju učenja. Rezultati metode ADAM iako su najslabiji po točnosti, jako su dobri po konzistentnosti što ukazuje da bi se bo-

ljim odabirom hiperparametara mogli dobiti kvalitetni rezultati. Momentum metoda je imala širok raspon vrijednosti s visokim maksimumom i najnižim minimumom, razlog tome je što vrijednost koju momentum nadodaje na standardni gradijentni spust može uzrokovati nagli pomak prema dobrom ili lošem rješenju. Gradijenti spust, momentum i ADAM su pokazali lošu tendenciju da se rezultati grupiraju u donjoj polovici. Nadam je postigao vrlo visoku točnost te su njegovi rezultati koncentriraniji u gornjoj polovici nego u donjoj. RMSProp se pokazao boljim od konkurentnog Nadama i po maksimalnoj, srednjoj točnosti, te raspodjeli točnosti u gornjoj polovici.

5. Zaključak

Optimizacija neuronskih mreža je nužan proces kako bi se dobili kvalitetni rezultati. Cilj ovoga rada je prikazati kako različiti odabiri hiperparametara utječu na brzinu učenja, krajnju točnost mreže te na primjeru prepoznavanja znamenki odraditi optimizaciju mreže. Rezultati su pokazali kako je za ovaj konkretan problem raspoznavanja ručno pisanih znamenki MNIST skupa podataka, sa trenutnim postavkama arhitekture i hiperparametara mreže optimalna metoda učenja RMSProp. Sve metode pokazale su bolja svojstva od učenja običnim gradijentnim spustom, izuzev ADAMa koji zahtjeva dodatnu optimizaciju. Treba uzeti u obzir da su ovi rezultati dobiveni sa relativno malim brojem koraka optimizacije, odnosno ispitivao se premali broj kombinacija parametara kako bi se s visokom sigurnošću odredio međusobni odnos metoda učenja te najbolja metoda. Bolju sposobnost učenja može se dobiti dodavanjem dodatnog konvolucijskog i *pooling* sloja (Deotte, 2018) te odabirom boljih hiperparametara mreže. Također performanse mreže se mogu poboljšati koristeći naprednije tehnike inicijalizacija filtara i težina između neurona (Philipp et al., 2016), (Doshi). Kako performanse konvolucijskih mreža jako ovise o arhitekturi same mreže, istražuju se i mogućnost korištenja genetskih algoritama kod stvaranja arhitektura (Sun et al., 2018).

LITERATURA

Chris Deotte. How to choose CNN architecture MNIST. 2018. URL https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist.

Neerja Doshi. Deep learning best practices (1) — weight initialization. URL https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94.

Casper Hansen. Optimizers explained. URL https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94.

Yann LeCun i Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Krähenbühl Philipp, Doersch Carl, Donahue Jeff, i Darrell Trevor. Data-dependent initializations of convolutional neural networks. *CoRR*, abs/1511.06856, 2016. URL http://arxiv.org/abs/1511.06856.

Sebastion Ruder. An overview of gradient descent optimization algorithms. URL https://ruder.io/optimizing-gradient-descent/.

Yanan Sun, Bing Xue, Mengjie Zhang, i Gary G. Yen. Automatically designing CNN architectures using genetic algorithm for image classification. *CoRR*, abs/1808.03818, 2018. URL http://arxiv.org/abs/1808.03818.

Sažetak

Konvolucijske neuronske mreže su model koji se često koristi u strojnom učenju zbog svoje prilagodljivosti. Kako bi se mreža dobro prilagodila određenom problemu potrebno optimizirati kombinaciju niza parametara koji će utjecati na sposobnost generalizacije i brzinu učenja. U ovom radu predstavljeni su metode koje se mogu koristiti i parametri koji se mogu namještati za vrijeme optimizacije neuronske mreže. Dodatno odrađena je optimizacija nad kombinacijama parametara te je analiziran krajnji rezultat pomoću vremenskih i kutijastih grafova.

Ključne riječi: konvolucijske neuronske mreže, optimizacijski postupci, parametri

Methods for optimising parameters of convolutional neural networks

Abstract

Convolutional neural networks are often used model in machine learning because of their adaptability. To make a network adapt to a certain problem, it is necessary to optimize the combination of a number of parameters which will affect the network's ability to generalize and its learning speed. This thesis presents various methods that can be used and parameters that can be adjusted during the optimization of the network. It also perfoms optimization and analyzes the end result using accuracy-time and box plots.

Keywords: convolutional neural networks, optimising procedures, parameters