SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 867

Biblioteka za specifikaciju i učenje neuronskih mreža

Damir Numić-Meša

Zagreb, svibanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 867

Pristupnik:	**Damir Numić-Meša (0036535035)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Marko Đurasević

Zadatak: **Biblioteka za specifikaciju i učenje neuronskih mreža**

Opis zadatka:

Proučiti koncept umjetnih neuronskih mreža, te varijanti mreža koje postoje s obzirom na njihovu arhitekturu. Istražiti postupke učenja neuronskih mreža temeljene na propagaciji unatrag, varijante aktivacijskih funkcija i funkcija gubitaka koje se mogu koristiti prilikom treniranja neuronskih mreža. Razviti biblioteku koja omogućuje specifikaciju neuronske mreže proizvoljne arhitekture, te omogućuje njezino treniranje na proizvoljnom problemu. Ocijeniti učinkovitost razvijene biblioteke korištenjem odabranih klasifikacijskih ili regresijskih problema. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Rok za predaju rada: 9. lipnja 2023.

SADRŽAJ

1. Uvod

Zamislimo da nam je dan zadatak u kojem na određenoj slici trebamo prepoznati objekt, npr. napisanu brojčanu znamenku. Zadatak zvuči trivijalno za čovjeka i velika većina ljudi uspješno će ga izvršiti. Potencijalne greške dolazit će zbog varijacije u rukopisima među znamenkama, no to nam u konačnici ne bi trebalo predstavljati veliki problem. Zamislimo pak sada da nam je dan isti takav zadatak, no ovaj put naše rješenje mora biti u obliku programa, tj. programskog koda. Kako bi sada pristupili rješavanju problema?

Slike na računalu prezentirane su u matričnom (ili u slučaju slika u boji, višematričnom) obliku, tj. svaki piksel slike prezentiran je određenom vrijednosti koju ćemo u našem hipotetskom problemu ograničiti na 0 ili 1 - tzv. binarne slike. Mogli bismo tada generirati sve moguće vrijednosti slika, tj. vrijednosti piksela na slici postavljenih ih na vrijednosti 0 ili 1. Svaki piksel slike može ili ne mora biti aktivan, što nas u konačnici dovodi do $2^{\text{visina} \cdot \text{širina}}$ različitih slika. Spremimo navedene slike za usporedbu i označimo ih s pripadnim brojevnim oznakama. Zadatak bi tada riješili tako da bi svaku učitanu sliku uspoređivali sa našom bazom podataka za podudaranje.

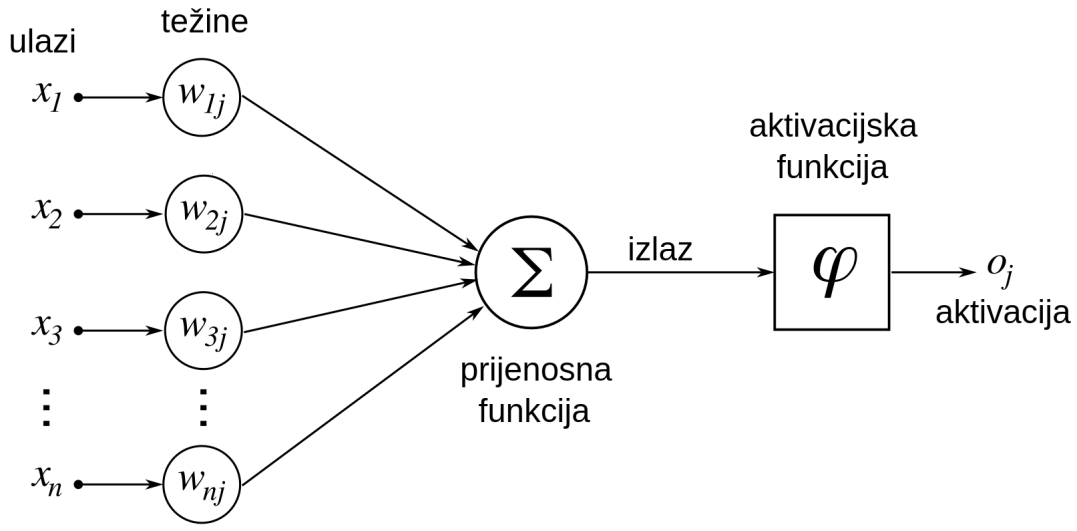
Problem s ovakvim pristupom dakako je kombinatorna eksplozija. Za malu sliku 10x10 piksela naš sustav trebao bi izgenerirati i pohraniti 2^{100} različitih slika. Ako svaki piksel slike zauzima 1B u memoriji računala dobivamo $2^{100} \cdot 10 \cdot 10 \cdot 1B = 1.268 \cdot 10^{32} B \approx 1.1210^{17}$ egzabajta, što je u potpunosti, memorijski, a i vremenski neizvedivo.

Umjetna neuronska mreža metoda je u grani umjetne inteligencije i računalnog učenja koja svoju motivaciju pronalazi upravo u ljudskome mozgu. Koristeći znanja linearne algebre i matematike objasniti ćemo kako neuronska mreža uči, izvest ćemo formule koje omogućavaju njeno učenje, te u konačnici razviti biblioteku koja korisnicima omogućava rješavanje jednostavnih klasifikacijskih problema.

2. Neuronska mreža

2.1. Neuron

Neuron predstavlja osnovnu gradivnu jedinicu neuronske mreže. Njegova struktura prikazana je na slici 2.1. U konačnici je to matematička funkcija koja se sastoji jednog/više ulaza i jednog izlaza nad kojima se provode transformacije. Svaki ulaz u neuron izaziva pobudu čiji je intenzitet određen koeficijentom za taj ulaz. Koeficijente pridružene ulazima nazivamo *težine*, dok izlaz iz neuron nazivamo *aktivacijom*. *Izlaz* predstavlja težinsku sumu umnoška težina i odgovarajućih ulaza $\sum_1^n x_i \cdot w_{ij}$, ili u matričnoj notaciji $\mathbf{W_j^T} \cdot \mathbf{x}$. Aktivacijsku funkciju zasada promatrajmo također kao transformaciju izlaza prijenosne funkcije (što ustvari i je), bez ulaženja u posebitosti njene transformacije.



Slika 2.1: Neuron

Uz gore navedenu strukturu neurona, ono što se još pronalazi je tzv. *bias*. Njega uobičajeno modeliramo tako da proširimo vektor incijalnih težina i ulazni vektor za

jedan:

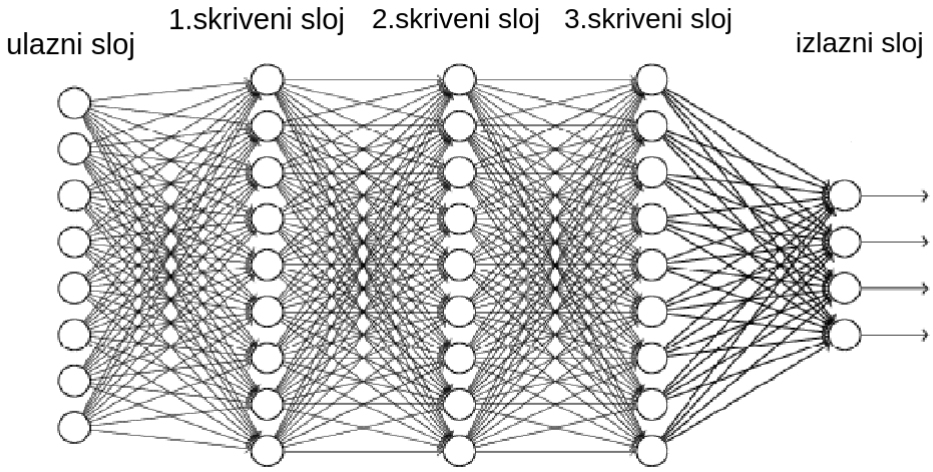
$$\mathbf{W_j} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ w_{n+1} = 1 \end{bmatrix} \qquad \mathbf{x_n} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ x_{n+1} = 1 \end{bmatrix} \tag{2.1}$$

2.1.1. Povezivanje više neurona

Kako jedan neuron sam po sebi uobičajeno nije dovoljan za rješavanje problema često ih povezujemo u višeslojne strukture, gdje aktivacija jednog neurona predstavlja ulaz u neurone narednih slojeva. Slojevi su međusobno potpuno povezani, te stoga ovi slojevi obično nose naziv potpuno povezani sloj. Vektor težina za i-ti sloj tada postaje matrica gdje redak matrice odgovara vektoru težina j-tog neurona tog sloja:

$$W_i = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} & w_{1(n+1)} \\ w_{21} & w_{22} & \dots & w_{2n} & w_{2(n+1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & \dots & w_{nn} & w_{n(n+1)} \end{bmatrix} \tag{2.2}$$

Sloj čije izlaze promatramo i uspoređujemo naziva se izlazni sloj, dok se unutarnji slojevi nazivaju skriveni slojevi. Primjer strukture s 3 skrivena sloja i izlaznim slojem dan je na slici 2.2.



Slika 2.2: Struktura više neurona s više slojeva

2.2. Aktivacijske funkcije

2.2.1. Motivacija

Potreba za dodatnom transformacijom izlaza javlja se zbog umnoška matrice težina i ulaza. Takav izlaz je linearna kombinacija elemenata ulaznog vektora i on može odgovarati rješavanju određenih tipova problema gdje je očekivana veza ulaza i izlaza linearna, npr. linearne regresije. U slučajevima kada je veza između ulaznih podataka i očekivanog izlaza nelinearne prirode potrebna nam je dodatna transformacija - aktivacijska funkcija. U sklopu ove biblioteke implementirane su četiri aktivacijske funkcije.

2.2.2. ReLU

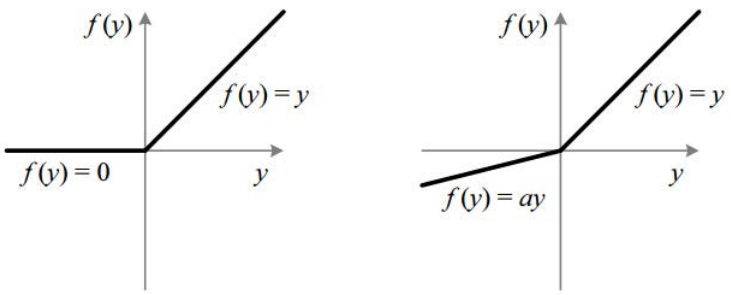
Rectified Linear Unit ili *ReLU* definirana je kao aktivacijska funkcija koja negativne vrijednosti izlaza neurona transformira u nulu, dok je za pozitivne vrijednosti funkcija identiteta:

$$\text{ReLU}(\mathbf{y}) = \max(\mathbf{0}, \mathbf{y}) \tag{2.3}$$

2.2.3. Leaky ReLU

Leaky Rectified Linear Unit ili *Leaky ReLU* varijacija je na prethodno definiranu ReLU funkciju, što je prikazano na slici 2.3. Transformacija pozitivnog izlaza neurona jednaka je kao i u ReLU funkciji, no negativne vrijednosti nisu preslikane u nulu već su skalirane faktorom α:

$$\text{LeakyReLU}(\mathbf{y}) = \begin{cases} \mathbf{y}, & \mathbf{y} > 0 \\ \alpha\mathbf{y}, & \mathbf{y} < 0 \end{cases} \tag{2.4}$$



Slika 2.3: ReLU i LeakyReLU aktivacijska funkcija

2.2.4. Sigmoida

Sigmoida je aktivacijska funkcija karakterističnog *S* oblika. Ulazne vrijednosti preslikane su u raspon $(0, 1)$ što odgovara matematičkoj definiciji funkcije vjerojatnosti:

$$\text{Sigmoid}(\mathbf{y}) = \frac{1}{1 + e^{-\mathbf{y}}} \tag{2.5}$$

2.2.5. Softmax

Softmax je aktivacijska funkcija koja također preslikava ulazne vrijednosti u raspon $(0, 1)$:

$$\text{Softmax}(\mathbf{y}) = \frac{e^{y_i}}{\sum_j^N e^{y_j}} \tag{2.6}$$

Za razliku od funkcije sigmoide, ovdje suma vjerojatnosti izlaznog vektora ima vrijednost 1.

2.3. Kako neuronska mreža uči?

Proces učenja neuronske mreže sastoji se od tri slijedne, međusobno povezane faze, koje ćemo u daljnjem tekstu analizirati:

- unaprijedna propagacija
- računanje pogreške
- unazadna propagacija

2.3.1. Unaprijedna propagacija - *forward propagation*

Unaprijedna propagacija ili *forward propagation* odnosi se na dio procesa računanja u kojem su podaci transformirani od neurona ulaznog sloja do neurona izlaznog sloja. Broj izlaznih i ulaznih neurona je varijabilan i u konačnici definiran problemom kojeg modeliramo. Unaprijedna propagacija definirana je izrazom:

$$\mathbf{izlaz} = aktivacija(\mathbf{W} \cdot \mathbf{ulaz}) \tag{2.7}$$

2.3.2. Funkcija gubitka

Funkcija gubitka ili funkcija pogreške definirana je kao funkcija koja uspoređuje izlaz neuronske mreže s očekivanim vrijednostima. Broj takvih funkcija je neograničen, no

mi ćemo se fokusirati na dvije, kategoričku unakrsnu entropiju (engl. *categorical cross entropy* - CCE) i srednju kvadratnu pogrešku (engl. *mean square error* - MSE) koje su i implementirane u sklopu ove biblioteke.

Kategorička unakrsna entropija - CCE

Za početak potrebno je definirati očekivanu strukturu izlaza neuronske mreže, a za primjer ćemo ponovno uzeti prethodno opisani problem prepoznavanja znamenki na čiji smo izlaz primjenili *softmax* aktivacijsku funkciju. Izlaz će biti oblika $N \times 1$ gdje se N odnosi na broj razreda kojima ulazna vrijednost može pripadati, što je u našem slučaju 10 razlčitih znamenki. Da bi usporedili takav izlaz s očekivanim izlazom potrebno je očekivani izlaz transformirati u isti oblik kao i izlaz iz neuronske mreže. Ono što dobijemo naziva se *one-hot encoded* prezentacija očekivanog izlaznog vektora koja je pogodna za korištenje CCE funkcije. Primjer *one-hot encoded* vektora dan je na slici 2.4.

$$\hat{\mathbf{y}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Slika 2.4: *One-hot encoded* vektora za broj 2

CCE funkcije je dalje definirana sljedećim izrazom:

$$\text{CCE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{n} y_i \log \hat{y}_i \tag{2.8}$$

gdje je $\hat{\mathbf{y}}$ izlaz neuronske mreže, a $\mathbf{y}$ *one-hot encoded* očekivana vrijednost.

Srednja kvadratna pogreška - MSE

Srednja kvadratna pogreška često je korištena funkcija pogreške koja mjeri prosječnu vrijednost kvadrata razlike izlaza neuronske mreže i očekivane vrijednosti izlaza. Definirana je sljedećim izrazom:

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i}^{N} (\hat{y}_i - y_i)^2 \tag{2.9}$$

2.3.3. Unazadna propagacija - *backpropagation*

Neuronska mreža definirana je kao kompozicija diferencijabilnih/kvazi-diferencijabilnih matematičkih funkcija koje transformiraju ulazne vrijednosti u izlaz definiran problemom. Postizanje minimalne vrijednosti pogreške biti će ostvareno za određenu kombinaciju vrijednosti matrica težina slojeva. Ono što želimo pronaći je koliki utjecaj mala promjena u vrijednosti težine nekog sloja ima na izlaz iz mreže, tj. $\frac{\partial izlaz}{\partial w_{ij}^{(l)}}$, gdje je w_{ij} dio matrice težina sloja l. Ako znamo da je najveći porast vrijednosti funkcije ostvaren za kretanje u smjeru njenog gradijenta, kretanje u smjeru negativnog gradijenta trebao bi nas dovesti u minimum funkcije. Procedura je sljedeća:

- za dobiveni ulaz u mrežu odredi izlaz svih slojeva
- za neurone i posljednjeg sloja l odredi iznos pogreške koristeći derivaciju funkcije gubitka (npr. MSE) L:

$$\delta_i^l = 2 \cdot (\hat{y}_i - y_i) \tag{2.10}$$

- za sloj $(l-1)$, ako je riječ o sloju aktivacijske funkcije izračunaj:

$$\delta_i^{(l-1)} = \frac{\partial f(x_i)}{\partial x_i} \cdot \delta_i^{(l)} \tag{2.11}$$

  inače izračunaj:

$$\delta_i^{(l-1)} = \sum_i w_{ij}^{(l)} \cdot \delta_i^{(l)} \tag{2.12}$$

$$\frac{\partial L}{\partial w_{ij}^{(l-1)}} = x_j^{(l-1)} \cdot \delta_i^{(l)} \tag{2.13}$$

- algoritmom iz poglavlja 4 prilagodi matricu težina $\mathbf{W}$.

Verifikacija implementacije unazadne propagacije

Kako bi se uvjerili u implementacijsku korektnost *backpropagation* algoritma, iskoristit ćemo koncept provjere gradijenta (engl. *gradient checking*). Provjera se temelji na aproksimaciji derivacije funkcije centralnim konačnim diferencijama:

$$f^{'}(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{2.14}$$

Razlika između numeričkog gradijenta dobivenog aproksimacijom i gradijenta dobivenog *backpropagation* algoritmom u prosjeku je 10^{-10}.

3. Konvolucijska neuronska mreža

Kako neuronska mreža očekuje vektor na ulazu u potpuno povezani sloj, multidimenzionalne podatkovne oblike (kao što su npr. slike) potrebno je pretvoriti vektorski oblik i tek onda ih proslijediti dalje. Mreža će naučiti prepoznati primjerak takve slike i uspješno riješiti problem. No što se događa u situaciji kada želimo prepoznati objekt interesa na translatiranoj slici?

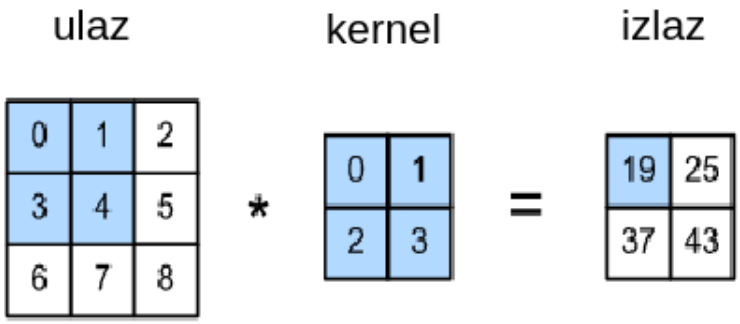
3.1. Lokalnost i geometrijska invarijantnost

Lokalnost je jedna od karakteristika slike koja nam govori o korelaciji susjednih regija piksela na slici. Na primjer, uzmimo sliku psa. Pikseli koji formiraju pseću njušku ili uho bit će lokacijski bliže jedan drugome u odnosu na druge piksele te će zajedno predstavljati prepoznatljive karakteristike objekta.

Geometrijska invarijantnost također je jedna od karakteristika slike i odnosi se na pojavu u kojoj rotacija, skaliranje i druge transformacije ne mijenjaju obilježja i informacije na slici, tj. ne mijenjaju njenju intepretabilnost. Nije svaka slika geometrijski invarijantna, no ovo svojstvo predstavlja jednu od temeljnih pojava koje konvolucijske neuronske mreže pokušavaju iskoristiti.

3.2. Konvolucija

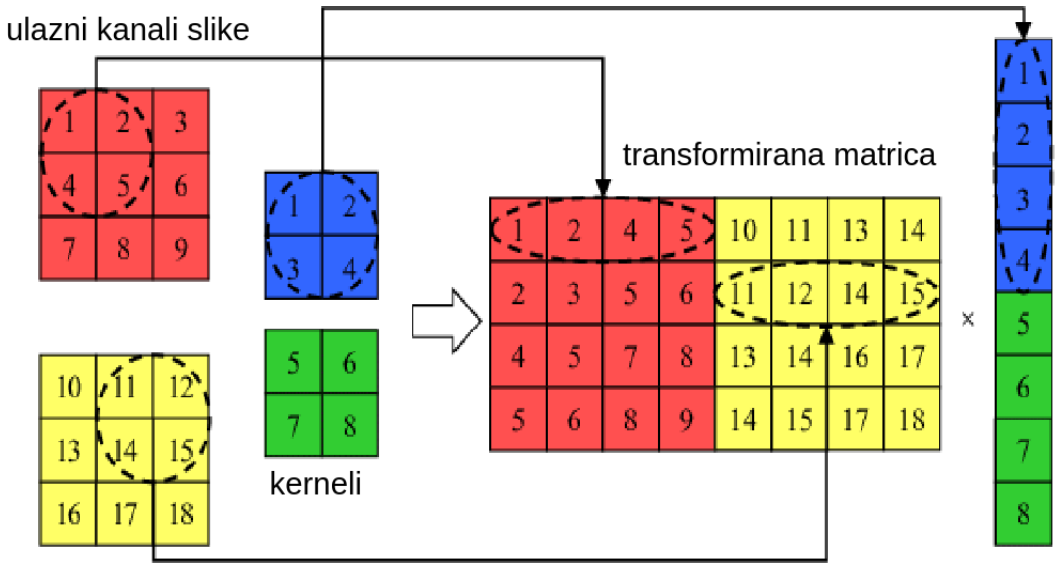
Konvolucijska neuronska mreža na ulaz dobiva 2D podatak i nizom operacija konvolucija transformira ju u podatak smanjenih dimenzija. Operacija konvolucije provodi se koristeći *kernel* - matricu koja se koristi za izoštravanje, zamagljivanje, detekciju rubova i mnoge druge svrhe, koji se pomiče po ulaznoj slici izvodeći umožak sa njenim elementima. Za ulaz veličine $H \times W$ piksela, izlaz će biti dimenzija $(H - K + 1) \times (W - K + 1)$ piksela te će sadržavati važne identifikacijske karakteristike dobivenog ulaza.



Slika 3.1: Primjer konvolucije nad 3x3 ulazom

3.3. Unaprijedna propagacija - *forward propagation*

U našem slučaju konvoluciju je potrebno provoditi nad velikim brojem podataka koje se sastoje od 3 dimenzije (kanal, visina, širina), dok su težine reprezentirane u obliku (izlazni kanali, ulazni kanali, visina, širina). Radi ubrzanja izvođenja konvolucije ulazne podatke transformiramo tako da prikažemo sve lokacije na kojima se *kernel* može nalaziti, te zatim izvodimo jedno matrično množenje sa transformiranom slikom i transformiranim težinama. Navedena metoda naziva se *im2col - image to column*, sa svojim reverznim postupkom *col2im - column to image*, te značajno ubrzava proces provođenja konvolucije. Primjer transformacije dan je na sljedećoj slici:



Slika 3.2: im2col transformacija

3.4. Unazadna propagacija - *backpropagation*

Slično kao i u slučaju unazadne propagacije u potpuno povezanim slojevima, i ovdje je potrebno dobiti ovisnost izlaza sloja o ulazu i vrijednostima *kernela*. Oznaka za konvoluciju je *. Procedura je sljedeća:

– za sloj $(l-1)$ izračunaj:

$$\delta_{\mathbf{i}}{}^{(l-1)} = \sum_i \sum_j \delta_{\mathbf{i}}{}^{(l)} * \mathbf{W_{ij}}{}^{(l)} \tag{3.1}$$

$$\frac{\partial L}{\partial \mathbf{W_{ij}}{}^{(l-1)}} = \mathbf{x_j}{}^{(l-1)} * \delta_{\mathbf{i}}{}^{(l)} \tag{3.2}$$

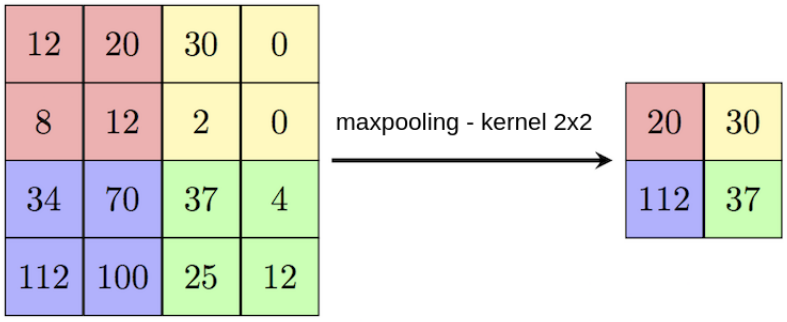
gdje je $\mathbf{W_{ij}}$ matrica $h \cdot w$ *kernela* rotirana $180°$.

– algoritmom iz poglavlja 4 prilagodi *kernele* $\mathbf{W_{ij}}$.

Backpropagation algoritam također je implementiran korištenjem jednog matričnog množenja, kao i u slučaju unaprijedne propagacije, metodama *im2col* i *col2im*.

3.5. Smanjivanje dimenzija podataka - *maxpooling*

Kako bi iz dobivene konvolucije izvukli aktivne regije piksela, provodimo dodatnu transformaciju koja se naziva *maxpooling*. Nad regijom ulaza veličine *kernela*, uzimamo maksimalne vrijednosti te zatim pomičemo *kernel* na susjednu regiju. Rezultat je podatak smanjenih dimenzija gdje su prisutni samo najaktivniji pikseli izvornog ulaza. Procedura je vrlo korisna jer smanjuje broj parametara potrebnih u narednim slojevima, tj. kompleksnost neuronske mreže, i time posljedično, vremensku složenost provođenja unaprijedne i unazadne propagacije. Primjer *maxpooling* operacije nad 4x4 ulaznim podatkom i veličinom *kernela* 2x2 dan je sljedećom slikom:



Slika 3.3: MaxPooling operacija

4. Optimizatori

Pojam optimizatora odnosi se na algoritam koji definira prilagođavanja matrice težina slojeva neuronske mreže. Postoji ih nekoliko, no u sklopu ove biblioteke implementirana su dva poznata i često korištena algoritma, SGD i Adam.

4.1. SGD

Definicija pojma SGD (engl. *Stohastic Gradient Descent*) je u današnjoj nomeklaturi neuronskih mreža i strojnog učenja nekonzistentna. Negdje se pojam odnosi na proces prilagođavanja težina na temelju jednog primjerka podatka za učenje, dok se drugdje taj isti pojam smatra kao općeniti pristup prilagođavanju matrice težina (neovisno o broju primjeraka podataka za učenje). U našem kontekstu promatrat ćemo SGD kao općeniti pristup ažuriranju matrice težine.

Za definiranu stopu učenja i dobiveni $\frac{\partial L}{\partial w^{(l)}}$ u koraku t sloja l težine su prilagođene sljedećim izrazom:

$$w^{l}_{(t+1)} = w_t - \alpha \frac{\partial L}{\partial w^{(l)}} \tag{4.1}$$

4.2. Adam

Adam optimizator nastao je kao kombinacija SGD-a i RMSP-a (engl. *Root Mean Square Propagation*). Ime mu dolazi od korištenja estimacija prvog i drugog momenta gradijenta prilikom prilagođavanja težine neuronske mreže. Momenti su dani sljedećim izrazima:

$$m^{(l)}_{t+1} = \beta_1 * m^{(l)}_t + (1 - \beta_1) * \left(\frac{\partial L}{\partial w^{(l)}_t} \right) \tag{4.2}$$

$$v^{(l)}_{t+1} = \beta_2 * v^{(l)}_t + (1 - \beta_2) * \left(\frac{\partial L}{\partial w^{(l)}_t} \right)^2 \tag{4.3}$$

Parametri β_1 i β_2 kontroliraju stopu utjecaja prethodnih momenata, tj. njihov *decay-rate*.

Adam optimizator uvodi još korekciju *biasa* (koju momenti imaju prema vrijednosti 0 kada su β_1 i β_2 blizu 1) m_t^l i v_t^l koja je dana sljedećim izrazom:

$$\hat{m_{t+1}^{(l)}} = \frac{m_{t+1}^{(l)}}{1 - \beta_1} \tag{4.4}$$

$$\hat{v_{t+1}^{(l)}} = \frac{v_{t+1}^{(l)}}{1 - \beta_2} \tag{4.5}$$

U konačnici, za definiranu stopu učenja i dobiveni $\frac{\partial L}{w^{(l)}}$ u koraku t sloja l, težine su prilagođene sljedećim izrazom:

$$w_{(t+1)}^{(l)} = w_{(t)}^{(l)} - \alpha * \frac{\hat{m_{t+1}^{(l)}}}{\sqrt{\hat{v_{t+1}^{(l)}} + \epsilon}} \tag{4.6}$$

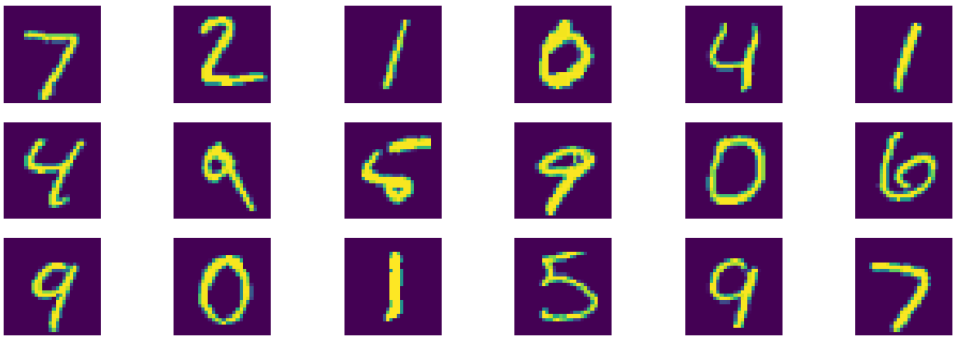
5. Klasifikacijski problemi

U prethodnim poglavljima objašnjena je struktura neuronske mreže, njene glavne komponente i način na koji neuronska mreža uči. Objašnjena je uloga konvolucije i konvolucijski pristup u izgradnji neuronskih mreža. U nastavku ćemo na nekolicini klasifikacijskih problema demonstrirati opisane funkcionalnsti koje nudi ova biblioteka i usporediti dobivene rezultate sa najboljim poznatim rezultatima. Konvencija prilikom definiranja potpuno povezanog sloja biti će navođenje ulaznog broja neurona i izlaznog broja neurona, dok će za konvolucijski sloj biti navođenje ulaznog broja kanala, izlaznog broja kanala, veličine kernela i vrijednosti *padding-a* koji se dodaje na ulazni podatak. Za aktivacijske funkcije podrazumijeva se da ne mijenjaju dimenzije podatka, te se izlaz i ulaz zato neće navoditi. Podaci iz skupa za testiranje koristit će se u procesu validacije te zato neće postojati zaseban skup za validaciju.

5.1. MNIST

MNIST podatkovni skup široko je poznat klasifikacijski skup podataka (spomenut još u uvodnom dijelu) na kojemu se često demonstriraju osnove učenja i sposobnost klasifikacije neuronske mreže. Podaci se sastoje od velikog broja slika crno-bijele boje ručno napisanih znamenki prikazanih na slici 5.1, veličine 28x28 piksela. Podijeljeni su u 10 međusobno isključivih razreda.

- skup za treniranje - 60000 slika
- skup za testiranje - 10000 slika



Slika 5.1: Primjer slika iz skupa za testiranje

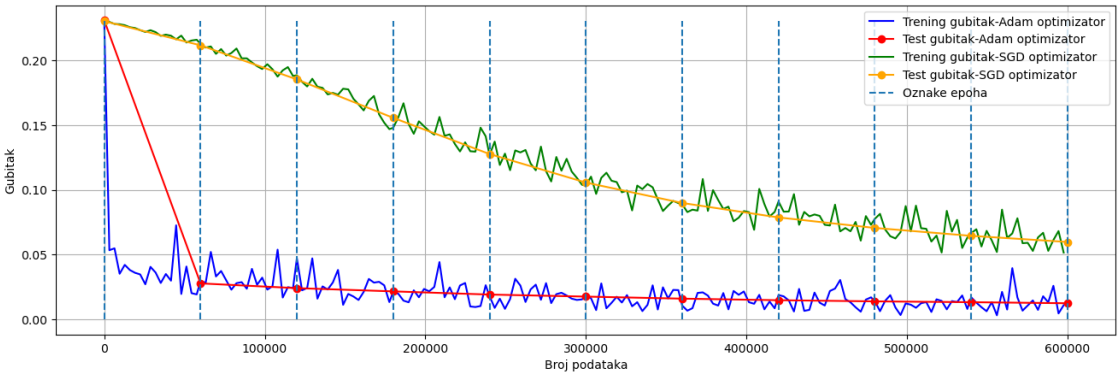
Naš problem pokušati ćemo riješiti koristeći dvije različite arhitekture neuronske mreže. Jedna će biti *obična* potpuno povezana neuronska mreža, dok će druga uključivati konvolucijske i *maxpooling* slojeve. Trenirat ćemo mreže kroz 10 epoha koristeći Adam i SGD optimizatore sa faktorom učenja od 0,001. Funkcija pogreške biti će CCE, tj. kategorička unakrsna entropija. Kako želimo usporediti osnovne implementacije algoritama, za SGD nećemo iskoristiti dodatne mogućnosti kao što su prilagođavanja težina korištenjem momentuma (engl. *momentum*) i/ili prigušivanja (engl. *dampening*), već ćemo samo zadati definiranu stopu učenja. Veličina *batch-a* biti će 64.

5.1.1. Rezultati 1. arhitekture

Tablica 5.1: Arhitektura potpuno povezane neuronske mreže

Naziv sloja	Konfiguracija	Aktivacijska funkcija
potpuno povezani sloj	(784,50)	ReLU
potpuno povezani sloj	(50,10)	Softmax

SGD optimizator postigao je lošije rezultate u odnosu na Adam algoritam, konvergirajući kasnije i sa većim iznosom pogreške u razdoblju od 10 epoha što je vidljivo na slici 5.2. Iako je riječ o *običnoj* neuronskoj mreži koja se sastoji samo od potpuno povezanih slojeva, postignuta je visoka klasifikacijska točnost od 96.36% korištenjem Adam optimizatora. Potencijalna poboljšanja ostvariva su korištenjem većeg broja neurona u skrivenom sloju i većem broju skrivenih slojeva.



Slika 5.2: Pogreška na skupu za treniranje i testiranje

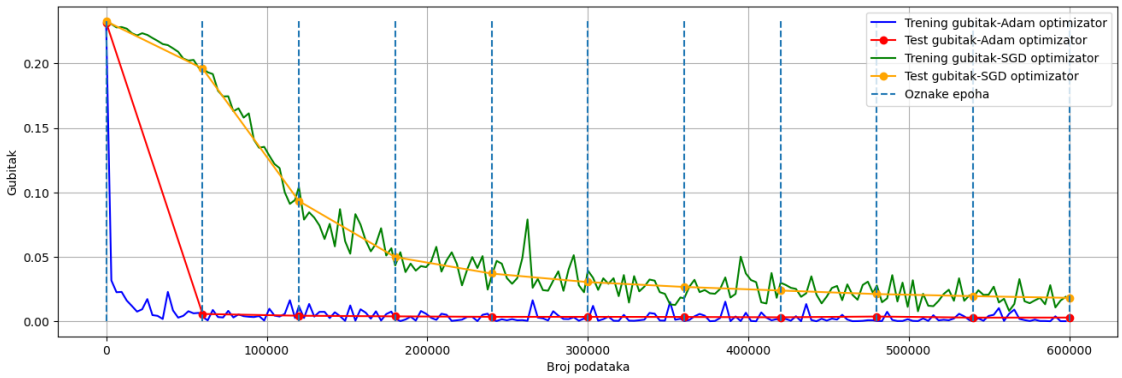
5.1.2. Rezultati 2. arhitekture

Tablica 5.2: Arhitektura konvolucijske neuronske mreže

Naziv sloja	Konfiguracija	Aktivacijska funkcija
2D konvolucija	(1,16,3x3,1)	ReLU
2D maxpooling	(2x2)	
2D konvolucija	(16,32,3x3,1)	ReLU
2D maxpooling	(2x2)	
2D konvolucija	(32,64,3x3,1)	ReLU
2D maxpooling	(2x2)	
potpuno povezani sloj	(576,10)	Softmax

Adam optimizator ponovno je postigao bolje rezultate u odnosu na SGD algoritam, što je vidljivo na slici 5.3, te je sa klasifikacijskom točnosti od 99.15% poboljšao rezultat potpuno povezane mreže iz prethode arhitekture. Najbolji poznati rezultat postiže 99.87% točnosti [5], no kompleksnost te mreže daleko je veća od trenutne. Ostvareni rezultat mogli bi se poboljšati složenijom arhitekturom i augmentacijom podataka.

Uočavamo da konvolucijska neuronska mreža pokazuje manje skokove u vrijednostima pogreške na skupu za treniranje i testiranje u odnosu na potpuno povezanu neuronsku mrežu, no veće razlike između dva navedena pristupa bit će vidljive na sljedećem, kompleksnijem problemu.

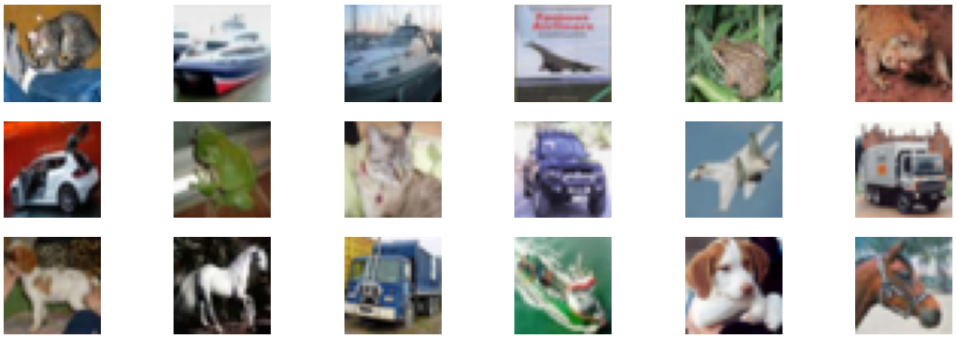


Slika 5.3: Pogreška na skupu za treniranje i testiranje

5.2. CIFAR-10

CIFAR-10 podatkovni skup sastoji se od 60000 slika u boji, 32x32 piksela, podijeljenih u 10 međusobno isključivih razreda: zrakoplov, automobil, ptica, mačka, jelen, pas, žaba, konj, brod i kamion, od kojih su neki prikazani na slici 5.4.

- skup za treniranje - 50000 slika
- skup za tesitranje - 10000 slika



Slika 5.4: Primjer slika iz skupa za testiranje

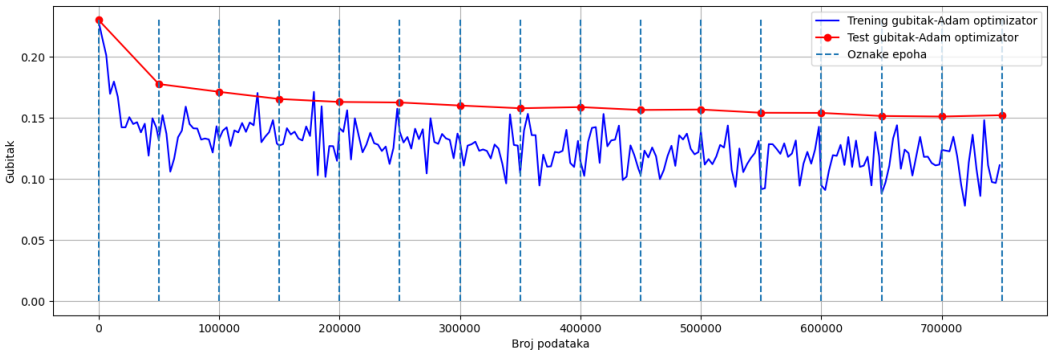
CIFAR-10 kompleksniji je problem od prethodno opisanog MNIST-a. Kako bi demonstrirali razliku u kompleksnosti koristiti ćemo obje arhitekture iz MNIST-problema (uz prilagodbu zbog različite veličine slika i povećanje broja neurona skrivenog sloja) uz Adam optimizator, koji je zasada pokazao najbolje performanse. Uz dvije navedene arhitekture konfigurirat ćemo još jednu složeniju na kojoj ćemo koristiti oba optimizatora. Mreže ćemo trenirati kroz 15 epoha uz stopu učenja od 0,001. Funkcija pogreške biti će CCE, tj. kategorička unakrsna entropija kao i u prethodnom problemu. Veličina *batch-a* biti će 64.

5.2.1. Rezultati 1. arhitekture

Tablica 5.3: Arhitektura potpuno povezane neuronske mreže

Naziv sloja	Konfiguracija	Aktivacijska funkcija
potpuno povezani sloj	(3072,256)	ReLU
potpuno povezani sloj	(256,10)	Softmax

Korištenje potpuno povezanih slojeva prikazanih u tablici 5.3 nije pokazalo dobre rezultate. Konačna klasifikacijska točnost na skupu za testiranje iznosi 46.31%. Na slici 5.5 je vidljivo da model pokazuje znatne oscilacije u iznosu pogreške na skupu za treniranje, te u konačnici stagnaciju pada pogreške, što može ukazivati na podnaučenost i/ili preveliku stopu učenja. Daljnje izmjene u broju skrivenih slojeva, broju neurona i aktivacijskoj funkciji između slojeva mogu rezultirati boljim rezultatima od ostvarenih.



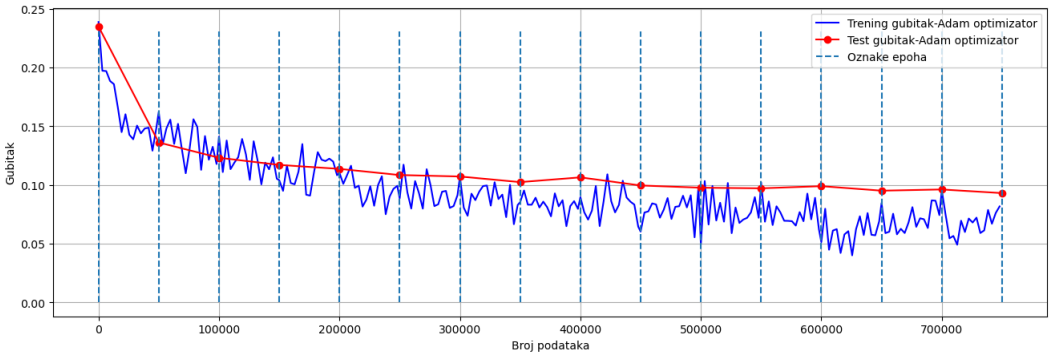
Slika 5.5: Pogreška na skupu za treniranje i testiranje

5.2.2. Rezultati 2. arhitekture

Tablica 5.4: Arhitektura konvolucijske neuronske mreže iz MNIST problema

Naziv sloja	Konfiguracija	Aktivacijska funkcija
2D konvolucija	(3,16,3x3,1)	ReLU
2D maxpooling	(2x2)	
2D konvolucija	(16,32,3x3,1)	ReLU
2D maxpooling	(2x2)	
2D konvolucija	(32,64,3x3,1)	ReLU
2D maxpooling	(2x2)	
potpuno povezani sloj	(1024,10)	Softmax

Iako se arhitektura iz tablice 5.4 pokazala uspješnom na klasifikaciji MNIST podatkovnog skupa, ovdje nije ostvarila željene rezultate u razdoblju od 10 epoha. Klasifikacijska točnost na skupu za testiranje iznosila je 67.75%. Uočavamo da vrijednosti pogreške na skupu za testiranje i treniranje, prikazane na slici 5.6, nisu imale značajnu promjenu nakon 9. epohe te zaključujemo da model nema dovoljnu kompleksnost za rješavanje problema. Arhitekturu je potrebno izmijeniti.



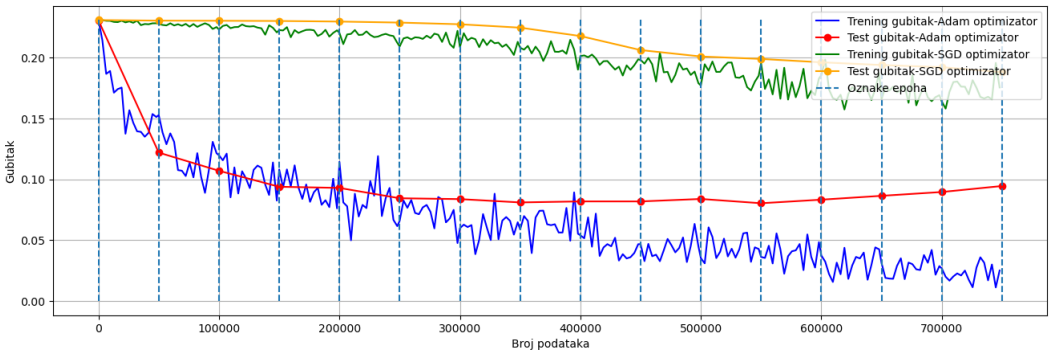
Slika 5.6: Pogreška na skupu za treniranje i testiranje

5.2.3. Rezultati 3. arhitekture

Tablica 5.5: Predložena kompleksnija konvolucijska neuronska mreža

Naziv sloja	Konfiguracija	Aktivacijska funkcija
2D konvolucija	(3,32,3x3,1)	ReLU
2D konvolucija	(32,32,3x3,1)	ReLU
2D maxpooling	(3x3)	
2D konvolucija	(32,64,3x3,1)	ReLU
2D konvolucija	(64,64,3x3,1)	ReLU
2D maxpooling	(3x3)	
2D konvolucija	(64,128,3x3,1)	ReLU
2D konvolucija	(128,128,3x3,1)	ReLU
2D maxpooling	(3x3)	
potpuno povezani sloj	(128,10)	Softmax

SGD optimizator ponovno se pokazao inferiornijim u odnosu na Adam u periodu od 10 epoha, što je vidljivo na slici 5.7. Novododana kompleksnija arhitektura opisana u tablici 5.5 postigla je 74.45% klasifikacijske točnosti na skupu za testiranje, u usporedbi sa najboljim poznatim rezultatom 99.7% [3]. Od 10. epohe na dalje vidljiva je divergencija pogreške između skupa za treniranje i testiranje što daje naznake prenaučenosti modela na trening podatke.

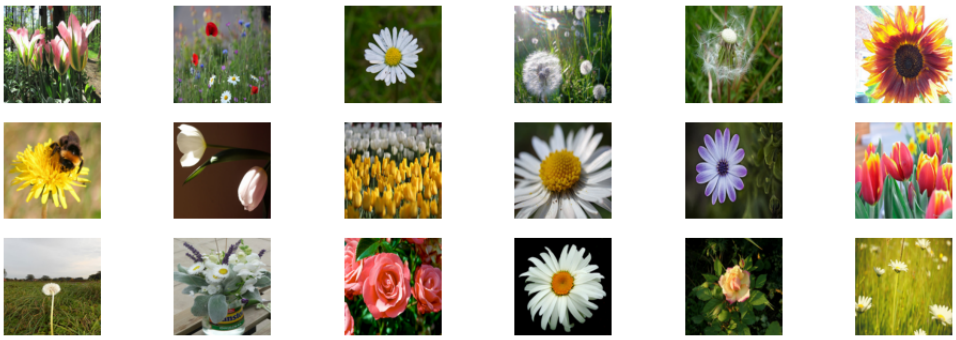


Slika 5.7: Pogreška na skupu za treniranje i testiranje

5.3. Flowers

Flowers skup podataka sadrži 4317 slika cvijeća u boji, podijeljenog u 5 razreda: kamilica, tulipan, ruža, suncokret i maslačak, od kojih su neki prikazani na slici 5.8. Svaki razred sadrži $\approx$ 800 slika maksimalne veličine 320x240 piksela. U našem slučaju slikama ćemo promijeniti veličinu na 128x128 piksela kako bi imali jednolik ulaz u neuronsku mrežu. Iz svakog razreda nasumično ćemo odabrati 250 primjeraka za testiranje, dok će ostatak biti upotrebljen za treniranje.

- skup za treniranje - 3067 slika
- skup za tesitranje - 1250 slika



Slika 5.8: Primjer slika iz skupa za testiranje

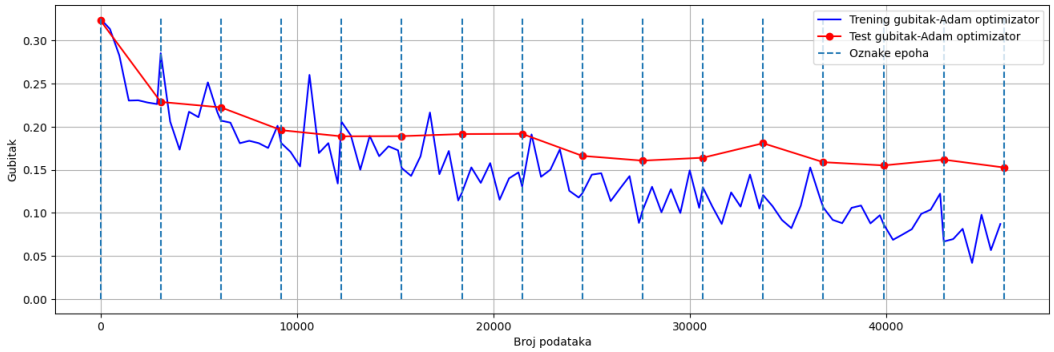
Navedeni problem najkompleksniji je do sada, slike su visoke rezolucije uz veliku varijabilnost među istim razredima te je prisutna mala količina podataka za treniranje. Koristiti ćemo samo jednu arhitekturu sa Adam optimizatorom, te trenirati mrežu na 15 epoha uz stopu učenja 0,001. Funkcija pogreške biti će CCE, tj. kategorička unakrsna entropija kao i u prethodnim problemima. Zbog memorijskih ograničenja treniranje ćemo odraditi s veličinom *batch-a* 48.

5.3.1. Rezultati arhitekture

Tablica 5.6: Predložena konvolucijska neuronska mreža

Naziv sloja	Konfiguracija	Aktivacijska funkcija
2D konvolucija	(3,24,3x3,1)	ReLU
2D maxpooling	(3x3)	
2D konvolucija	(24,48,3x3,1)	ReLU
2D maxpooling	(3x3)	
2D konvolucija	(48,96,3x3,1)	ReLU
2D maxpooling	(3x3)	
2D konvolucija	(96,192,3x3,1)	ReLU
2D maxpooling	(3x3)	
potpuno povezani sloj	(192,5)	Softmax

Model iz arhitekture opisane u tablici 5.6 ostvario je 72.96% klasifikacijske točnosti na skupu za testiranje. Na slici 5.9 primjećujemo relativno velike oscilacije u pogreški na skupu za treniranje, što može ukazivati na preveliku stopu učenja. Također uočavamo naznake divergencije iznosa pogreške na skupu za treniranje i skupu za testiranje, što daje naznake prenaučenosti modela na podatke za treniranje. S obzirom da je skup za treniranje male veličine, augmentacijom podataka možemo očekivati bolje rezultate od dobivenih.



Slika 5.9: Pogreška na skupu za treniranje i testiranje

Trajanje treniranja i testiranja

Za usporedbu vremena izvođenja treniranja iskoristiti ćemo PyTorch. Dva ispitana skupa podataka, MNIST i CIFAR10 su u njemu direktno dostupna te ćemo stoga performanse prikazati na njima. Upotrebljene arhitekture će koristiti Adam optimizator u obje biblioteke, te grafičku karticu dostupnu na računalu. Postignuta vremena su prikazana u tablici 5.7 te predstavljaju jedan ciklus izvođenja treniranja i testiranja. Uočljivo je da je vrijeme treniranja u PyTorchu znatno manja na svim arhitekturama, osim potpuno povezane mreže MNIST klasifikacijskog problema.

Tablica 5.7: Vrijeme treniranja i testiranja

Klasifikacijski problem	**Arhitektura**	**Trajanje**	**Trajanje - PyTorch**
MNIST	1.arhitektura	1min 39s	1min 33s
	2.arhitektura	3min 57s	1min 42s
CIFAR10	1.arhitektura	5min 45s	2min 18s
	2.arhitektura	5min 57s	2min 27s
	3.arhitektura	21min 54s	2min 43s

6. Zaključak

Danas, uz prisustvo brojnih alata (PyTorch, Tensorflow,...) gotovo svaki korisnik može izraditi model neuralne mreže za rješavanje raznoraznih problema, bez potrebe za poznavanjem intrinzične strukture njihove implementacije. Iako se isprva neuralne mreže mogu činiti kao sustavi kompleksne implementacijske specifičnosti, sposobni za rješavanje gotovo bilo kojeg problema, u pozadini se zapravo kriju jednostavni koncepti linearne algebre i matematike primjenjeni na elementarnu strukturu neuralne mreže sposobnu učenja - neuron. Razumijevanje ovih temeljnih koncepata omogućava korisnicima da prilagode i optimiziraju svoje modele, te im pomaže u rješavanju specifičnih problema koji nisu obuhvaćeni gotovim rješenjima. To je ujedno i jedan dio motivacije za izgradnju ove biblioteke - približiti pozadinu razvoja jednog ovakvog alata.

Iako konačna implementacija ne sadrži ostvarenja koncepata regularizacije (*dropout* slojevi i *batch* normalizacija), i dalje smo s relativnom uspješnosti riješili prezentirane klasifikacijske probleme. Općepoznati MNIST skup podataka nije predstavljao velik klasifikacijski problem, međutim, kod zahtjevnijih skupova podataka kao što su CIFAR10 i Flowers, susreli smo se s problemom prenaučenosti i povećanom vremenskom kompleksnosti treniranja modela sposobnog za klasifikaciju. Navedeni problemi imaju svoja rješenja korištenjem spomenute regularizacije, augmentacije podataka, drugih optimizatora, te poboljšavanja vremenske i prostorne složenosti algoritama učenja, od kojih svi predstavljaju potencijalni implementacijski dodatak u nastavku razvoja ove biblioteke.

LITERATURA

[1] Backpropagation. URL https://en.wikipedia.org/wiki/Backpropagation.

[2] The CIFAR-10 dataset, . URL https://www.cs.toronto.edu/~kriz/cifar.html.

[3] Image Classification on MNIST, . URL https://paperswithcode.com/sota/image-classification-on-cifar-10.

[4] CNN BackPropagation. URL https://deeplearning.cs.cmu.edu/F21/document/recitation/Recitation5/CNN_Backprop_Recitation_5_F21.pdf.

[5] Image Classification on MNIST. URL https://paperswithcode.com/sota/image-classification-on-mnist.

[6] Akash Ajagekar. Adam, 2021. URL https://optimization.cbe.cornell.edu/index.php?title=Adam.

[7] Raúl Gómez Bruballa. Understanding categorical cross-entropy loss, 2018. URL https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[8] Dariel Dato. MNIST in CSV, 2018. URL https://www.kaggle.com/datasets/oddrationale/mnist-in-csv.

[9] Jefkine Kafunah. Backpropagation in convolutional neural networks, 2016. URL https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/.

[10] Simeon Kostadinov. Understanding backpropagation algorithm, 2019. URL https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd.

[11] Alexandar Mamaev. Flowers Recognition, 2020. URL https://www.kaggle.com/datasets/alxmamaev/flowers-recognition.

[12] Michael Nielsen. How the backpropagation algorithm works. URL http://neuralnetworksanddeeplearning.com/chap2.html.

Biblioteka za specifikaciju i učenje neuronskih mreža

Sažetak

Neuralne mreže univerzalni su aproksimatori funkcija te stoga primjenjivi na rješavanja gotovo svakog oblika problema. U ovom završnom radu obraditi će se način na koji neuralne mreže uče, uvesti konvoluciju u pristup učenju i pokazati rezultate implementiranih algoritama na nekoliko različitih skupova podataka.

Ključne riječi: unazadna propagacija, unaprijedna propagacija, neuron, neuralna mreža, konvolucija, konvolucijska neuralna mreža, MNIST, CIFAR10, Adam, SGD

Library for specification and training of neural networks

Abstract

Neural networks are universal approximators and as such applicable on almost any problem. This bachelor thesis will process how neural networks learn, introduce convolution in its learning process and show results achieved by implemented algorithms on a few different datasets.

Keywords: backpropagation, forward propagation, neuron, neural network convolution, convolutional neural network, MNIST, CIFAR10, Adam, SGD