Primjena dubokog učenja u analizi rezultata pretraga koljena magnetskom rezonancom

Jaganjac, Danijela

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:647855

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-22**

mathos

Repository / Repozitorij:

Repository of School of Applied Mathematics and Informatics

zir.nsk.hr

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku

Danijela Jaganjac

Primjena dubokog učenja u analizi rezultata pretraga koljena magnetskom rezonancom

Diplomski rad

Osijek, 2019.

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku

Danijela Jaganjac

Primjena dubokog učenja u analizi rezultata pretraga koljena magnetskom rezonancom

Diplomski rad

Voditelj: izv. prof. dr. sc. Domagoj Matijević

Osijek, 2019.

Sadržaj

1. Uvod

Glavna tema ovog rada su neuronske i konvolucijske mreže. Neuronske mreže su efikasan alat za rješavanje različitih vrsta problema zbog mogućnosti prilagođavanja arhitekture modela, stoga se koriste u nadziranom učenju za regresiju i klasifikaciju. Temeljni dio u procesu učenja modela neuronskih i konvolucijskih mreža je optimizacija, no za razliku od standardnog problema optimizacije gdje je cilj pronaći optimalne parametre ovdje je cilj postići što veću točnost modela, a pretpostavka je da će upravo optimalni parametri tome doprinijeti. Stoga će teorijski dio rada pokriti osnovno znanje o neuronskim mrežama i optimizaciji.

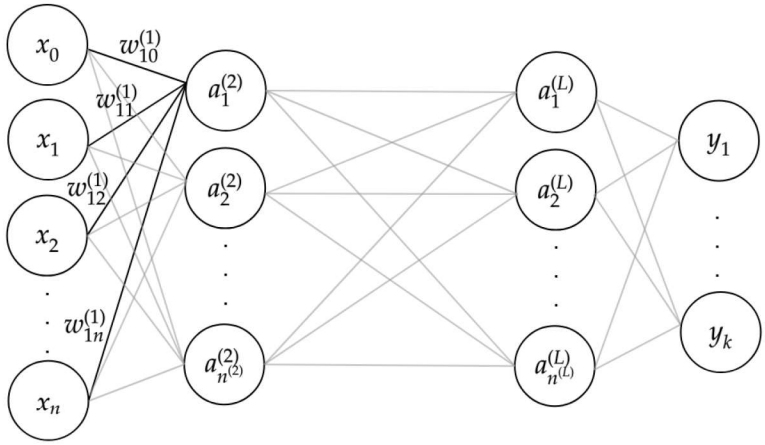
Računalni vid, odnosno zadaci klasifikacije slika i prepoznavanje objekata na slikama se mogu rješavati koristeći neuronske mreže. No kako broj parametara ovisi o arhitekturi i o veličini podataka, klasifikacija slika neuronskim mrežama dovodi do porasta broja parametara i računanje postaje vremenski i memorijski zahtjevno.
Rješenje je tada napraviti arhitekturu koja smanjuje broj parametara. Tu se pojavljuju konvolucijske mreže, čija je upotreba usmjerena na klasifikaciju slika. Kreiranje kvalitetnog modela konvolucijske mreže je netrivijalno zbog same arhitekture i zbog procesa učenja koji se mora provesti na velikom broju slika. Zbog toga se puno znanstveno - istraživačkih radova bavilo tom problematikom što je rezultiralo s nekoliko prepoznatljivih imena modela konvolucijskih mreža poput AlexNet-a, ResNeT-a, GoogLeNet-a i dr. U radu će biti objašnjena arhitektura konvolucijskih mreža i proces klasifikacije slika te predstavljen model AlexNet-a.

Razni modeli neuronskih i konvolucijskih mreža su razvijeni za analiziranje i predviđanje u bankarstvu, marketingu, meteorologiji i slično. Klasifikacija je vrlo primjenjiva u medicini gdje se na temelju podataka o pacijentima i bolestima može automatizirati proces davanja dijagnoze. Osim toga, u medicini se primjenjuje prepoznavanje uzoraka na slikama tako da se prepoznaju i klasificiraju medicinske slike izrađene tehnikama poput CT-a, rendgenskog snimanja i magnetske rezonance. Svrha takvog klasifikatora je pomoći liječnicima prilikom interpretacije snimki u smislu davanja brzih dijagnoza i lokalizacije oštećenih tkiva. U ovom radu će biti predstavljen problem klasifikacije snimki s magnetske rezonance koljena. Kako postoje već radovi koji su se bavili istim problemom, prvo ćemo predstaviti njihova rješenja na temelju kojih je implementirano i vlastito rješenje. Cilj ovog projekta je napraviti model koji postiže visoku točnost i pokazati kako se može upotrebljavati u svakodnevici pri dijagnosticiranju oštećenja koljena. U radu će biti opisani korišteni podaci, metode i dobiveni rezultati.

2. Neuronske mreže

2.1 Osnovni koncept neuronske mreže

U ovom poglavlju će biti objašnjen pojam neuronskih mreža.
Arhitektura modela neuronskih mreža je prikazana na slici 1. Neuronska mreža se sastoji od slojeva, a svaki sloj sadrži aktivacijske jedinice. Prvi i zadnji sloj su ulazni podaci, odnosno izlazne vrijednosti. Svi slojevi između se nazivaju skrivenim slojevima. Svaka jedinica u skrivenim slojevima je linearna kombinacija jedinica iz prethodnih slojeva pa se tako dobiju potpuno povezani slojevi. Svakoj aktivacijskoj jedinici je pridružena aktivacijska funkcija.



Slika 1: Grafički prikaz arhitekture neuronske mreže koja podatak veličine n transfomira kroz L slojeva i pridružuje predikcije za pripadanje jednoj od k klasa

Oznaka za aktivacijsku funkciju j-te jedinice u l-tom sloju je $a_j^{(l)}$. Formalno su aktivacijske funkcije prvog sloja zapravo ulazni podaci, $a_i^{(1)} = x_i, \forall i = 1, ..., n^{(1)}$. Stoga računanje aktivacijskih funkcija započinje u drugom sloju, a osnovni model za računanje je

$$a_j^{(2)} = \sum_{i=1}^{n^{(1)}} w_{ji}^{(1)} x_i + w_{j0}^{(1)},$$

gdje je $\mathbf{x} = x_1, ...x_{n^{(1)}}$ ulazni podatak, parametri $w_{ji}^{(1)}$ su težine koje pripadaju prvom sloju, a parametar $w_{j0}^{(1)}$ je pomak (*engl.* bias). Nadalje, svaka izračunata vrijednost $a_j^{(l)}$ se dalje transformira nekom nelinearnom, diferencijabilnom aktivacijskom funkcijom h

$$z_j^{(2)} = h(a_j^{(2)}).$$

Vrijednosti u sljedećim slojevima se analogno računaju. Općenito vrijednost aktivacijske funkcije k-te jedinice u l-tom sloju je linearna kombinacija svih jedinica iz $l-1$ sloja,

$$a_k^{(l)} = \sum_{j=1}^{n^{(l)}} w_{kj}^{(l-1)} z_j^{(l-1)} + w_{k0}^{(l-1)}.$$

Uobičajeno je uvesti dodatnu varijablu $x_0 := 1$ tako da se pomak apsorbira u vektor težina $\mathbf{w}$ pa aktivaciju u bilo kojem sloju možemo zapisivati kao

$$a_k^{(l)} = \sum_{i=0}^{n^{(l)}} w_{ji}^{(l-1)} z_j^{(l-1)}.$$

U posljednjem sloju, aktivacijske jedinice se preslikavaju u izlazne vrijednosti neuronske mreže. Aktivacijska funkcija se bira ovisno o podacima, odnosno prirodi problema koji se rješava. Vrijednosti izlaznih vrijednosti s obzirom na odabranu funkciju σ su tada

$$y_k = \sigma(\sum_{j=0}^{n^{(L)}} w_{kj}^{(L)} z_j^{(L)}).$$

Najčešći izbor za aktivacijsku funkciju su

- Sigmoid
- Softmax
- Tanh
- ReLU

Funkcija sigmoid svaki realni broj preslika u interval $[0, 1]$, stoga se koristi za predikciju vjerojatnosti. Definirana je sa

$$\sigma : \mathbb{R} \to [0, 1], \quad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

Funkcija softmax je generalizacija za slučaj multinomne klasifikacije i definirana je s

$$\Sigma : \mathbb{R}^K \to \mathbb{R}^K, \ \sigma(a)_j = \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}}, \ za \ j = 1, ..., K.$$

Tangens hiperbolni, odnosno tanh funkcija je definirana s

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Rectified Linear Unit, odnosno ReLU je nelinearna, aktivacijska funkcija definirana na sljedeći način

$$ReLU : \mathbb{R} \to [0, \infty], \ ReLU(x) = max\{0, x\}.$$

2.1.1 Proces učenja neuronske mreže

Podatke nad kojima se provodi proces učenja je uobičajeno podijeliti na tri dijela - skup podataka za treniranje, skup podataka za validaciju i skup podataka za testiranje. Kako ovakva vrsta učenja pripada nadziranom učenju, za svaki podatak je potrebno imati njegovu oznaku. Neka je tada $\mathbf{x}$ vektor ulaznih podataka i t njemu pripadna oznaka. Propuštanjem vektora $\mathbf{x}$ kroz neuronsku mrežu s težinama $\mathbf{w}$ dobiva se izlazna vrijednost $y(\mathbf{x}, \mathbf{w})$. Nadalje u tekstu ćemo istoznačno koristiti oznake $y(\mathbf{x}, \mathbf{w}) = y$.

Mjera odstupanja izračunate vrijednosti od točne vrijednosti je funkcija pogreške u ovisnosti o težinama $E(\mathbf{w})$. Cilj je optimizirati funkciju $E(\mathbf{w})$, odnosno pronaći takve parametre $\mathbf{w}$ u kojma funkcija E postiže globalni minimum. U nastavku će prvo biti pokazan odabir funkcija pogreške za probleme klasifikacije, a onda objašnjen postupak minimizacije istih. Vrijednosti izlaznog sloja mreže $y(\mathbf{x}, \mathbf{w})$ moguće je interpretirati kao vjerojatnosti pridruživanja stvarne oznake t ulaznom podatku $\mathbf{x}$ s obzirom na težine $\mathbf{w}$. Pretpostavimo da je zadan slučaj binarne klasifikacije. Ulazni vektor $\mathbf{x}$ može pripadati klasi C_1 i tada mu je oznaka $t = 1$, a u suprotnom pripada klasi C_0 i oznaka je $t = 0$. Pretpostavimo da izlazna jedinica sadrži sigmoid aktivacijsku funkciju, tada je izlazna vrijednost s obzirom na aktivacijske jedinice a

$$y(\mathbf{x}, \mathbf{w}) = \sigma(a) = \frac{1}{1 + e^{-a}}.$$

Za tako dobivene vrijednosti vrijedi $0 \leq y(\mathbf{x}, \mathbf{w}) \leq 1$ i interpretiraju se kao uvjetne vjerojatnosti za realizaciju oznake $t = 1$, dok je vjerojatnost za suprotni događaj $t = 0$ onda $1 - y(\mathbf{x}, \mathbf{w})$. Dakle, ulaznim podacima pridružene oznake t se modeliraju kao slučajne varijable iz Bernoullijeve distribucije s mogućim realizacijama $\{0, 1\}$ i pripadnim vjerojatnostima $\{1 - y(\mathbf{x}), y(\mathbf{x})\}$ što se može zapisati kao

$$p(t|\mathbf{x}, \mathbf{w}) = y(\mathbf{x}, \mathbf{w})^t (1 - y(\mathbf{x}, \mathbf{w}))^{1-t}.$$

Djelovanjem s negativnim logaritmom na prethodni izraz dobije se negativna log - izglednost pomoću koje se definira funkcija pogreške

$$-\ln(p(\mathbf{t}|\mathbf{w})) = -t \ln(y) - (1 - t) \ln(1 - y).$$

Ako pretpostavimo da imamo skup od N podataka, negativna log- izglednost se naziva funkcija unakrsne entropije (*engl.* cross-entropy function) i ima sljedeći oblik

$$E(\mathbf{w}) = -\sum_{i=1}^{N} t_n \ln(y_n) + (1 - t_n) \ln(1 - y_n).$$

U multinomnoj klasifikaciji, svaki podatak pripada jednoj od $K > 2$ međusobno disjunktnih klasa. Ulaznom podatku $\mathbf{x}$ je pridružena oznaka klase $\mathbf{t} = (t_1, ..., t_K)$ koja je kodirana na sljedeći način

$$t_k = \begin{cases} 1 & \text{ako } \mathbf{x} \textit{ pripada klasi } k \\ 0 & \text{inače} \end{cases}.$$

Zadnji sloj u neuronskoj mreži sadrži K aktivacijskih jedinica. Kao i kod binarne klasifikacije, na te aktivacijske jedinice se djeluje s odgovarajućom nelinearnom funkcijom kako bi se dobilo vrijednosti izlaznih varijabli $y_k(\mathbf{x}, \mathbf{w})$. Aktivacijska funkcija koja se primjenjuje je softmax funkcija općenito definirana s

$$\Sigma : \mathbb{R}^K \to \mathbb{R}^K, \ \sigma(a)_j = \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}}.$$

Vrijednosti $y_k = y_k(\mathbf{x}|\mathbf{w}), \ za \ k = 1, ..., K$ dobivene primjenom softmax funkcije na aktivacijske jedinice u zadnjem sloju se interpretiraju kao vjerojatnost pripadanja ulaznog vektora $\mathbf{x}$ klasi k. Može se pokazati kako je $0 \leq y_k \leq 1$ i $\sum_{k=1}^{K} y_k = 1$. Na kraju, funkcija pogreške računa se kao

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w})$$

2.1.2 Određivanje optimalnih parametara i propagacija unatrag

Određivanje optimalnih parametara, odnosno težina $\mathbf{w}$ je problem minimizacije funkcije pogreške E. Funkcije pogreške navedene u prethodnom poglavlju su diferencijabilne funkcije nad kojima se mogu primijeniti neki rezultati iz numeričke matematike. Općenito se problem višedimenzionalne minimizacije može riješiti iterativnim procesom oblika

$$w_{k+1} = w_k + \alpha p_k,\ k = 0, 1, ...,$$

gdje k označava korak, w_0 je početna aproksimacija za minimum, parametar $\alpha > 0$ se naziva duljina koraka, a p_k vektor smjera kretanja od točke w_k u točku w_{k+1}. Izbor vektora smjera i duljine koraka ovisi o metodi koja se koristi.
Gradijentna metoda za minimizaciju funkcije pogreške $E : \mathbb{R}^n \to \mathbb{R}$ definira se kao

$$w_{k+1} = w_k - \alpha \nabla E(w_k),\ k = 0, 1, ...$$

U svakoj iteraciji je potrebno ponovno izračunati vrijednost funkcije E, odnosno evaluirati funkciju na cijelom trening skupu. Kako bi se to izbjeglo, funkcija pogreške se može računati kao suma funkcija pogreške za svaki ulazni podatak

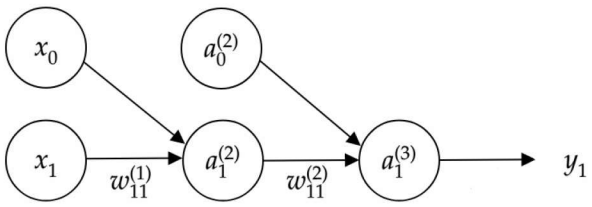
$$E(w) = \sum_{n=1}^{N} E_n(w).$$

Pripadna iterativna metoda za računanje minimuma funkcije se tada računa uzastopno za svaki ulaz x_n ili za nasumično odabrani x_n

$$w_{k+1} = w_k - \alpha \nabla E_n(w_k),\ k = 0, 1, ...$$

Dakle, jedan pristup računanju je korištenjem čitavog skupa podataka odjednom, s druge strane računanje se može izvršiti koristeći jedan po jedan podatak što se naziva pojedinačnim (*engl.* online) ili stohastičkim (*engl.* stochastic) pristupom. Treći pristup je odabiranjem manjih grupa podataka (*engl.* mini-batch).

Sljedeći cilj je pronaći efikasnu tehniku za evaluaciju gradijenta funkcije pogreške E. Kako bismo to napravili, uvesti ćemo potrebnu teoriju i oznake na primjeru, zatim na općenitom modelu neuronske mreže.



Slika 2: Jednostavni primjer modela neuronske mreže koja ima dva sloja i jednu aktivacijsku jedinicu u svakom sloju.

Neka je zadan jednostavni model neuronske mreže kao na slici 2. Neuronska mreža se sastoji od dva sloja, svaki sloj sadrži po jednu aktivacijsku jedinicu i dodatno pomak. Svakom sloju je pridružena aktivacijska funkcija h. Zbog jednostavnosti pretpostavimo da je aktivacijska funkcija u izlaznom sloju identiteta, odnosno $y_k = a_k^{(3)}$, a funkcija pogreške definirana kao suma kvadrata odstupanja $E(w) = \frac{1}{2}\sum_n (y_n - t_n)^2$. Zadan je jedan ulazni podatak $\mathbf{x} = x_1$ i njegova oznaka t_1.
Na temelju do sada uvedenih formula, napravimo propagaciju unaprijed.

$$a_1^{(2)} = w_{11}^{(1)} x_1 + w_{10}$$

$$z_1^{(2)} = h(a_1^{(2)})$$

$$a_1^{(3)} = w_{11}^{(2)} z_1^{(2)} + w_{10}^{(2)}$$

$$y_1 = h(a_1^{(3)}) = a_1^{(3)}$$

Izračunajmo sada parcijalne derivacije funkcije E s obzirom na težine u prvom i drugom sloju $\frac{\partial E}{\partial w_{11}^{(2)}}$, $\frac{\partial E}{\partial w_{11}^{(1)}}$.
Primjenom lančanog pravila na funkciju pogreške

$$E(w) = \frac{1}{2}(y_1 - t_1) = \frac{1}{2}(a_1^{(3)} - t_1),$$

slijedi

$$\frac{\partial E}{\partial w_{11}^{(2)}} = \frac{\partial E}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial w_{11}^{(2)}} = \frac{\partial E}{\partial a_1^{(3)}} z_1^{(2)}.$$

Uvedemo oznaku $\delta^{(2)} := \frac{\partial E}{\partial a_1^{(3)}}$. Nadalje, za sljedeću parcijalnu derivaciju ponovnom primjenom lančanog pravila dobijemo

$$\frac{\partial E}{\partial w_{11}^{(1)}} = \frac{\partial E}{\partial a_1^{(3)}} \cdot \frac{\partial a_1^{(3)}}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial w_{11}^{(1)}},$$

iz čega slijedi

$$\frac{\partial E}{\partial w_{11}^{(1)}} = \delta^{(2)} \cdot w_{11}^{(2)} \cdot h'(a_1^{(2)}) \cdot x_1.$$

Uvedimo oznaku $\delta^{(1)} := \delta^{(2)} \cdot w_{11}^{(2)} \cdot h'(a_1^{(2)})$. Primijetimo kako smo parcijalne derivacije računali od zadnjeg sloja prema naprijed. Upravo je to glavna ideja algoritma propagacija unatrag. U nastavku će ovaj postupak biti naveden za općeniti slučaj neuronske mreže.

Neka je zadan općeniti model neuronske mreže kao u poglavlju 2.1. Za zadani ulaz $\mathbf{x}_n$ propagacijom unaprijed se računaju vrijednosti aktivacijskih jedinica do posljednjeg sloja $y_{nk} = y_k(\mathbf{x}_n, w)$. Sada je potrebno izračunati parcijalnu derivacije pripadne funkcije pogreške E_n s obzirom na težinu $w_{ji}^{(l)}$. Funkcija E_n ovisi o težini $w_{ji}^{(l)}$ samo preko aktivacije $a_j^{(l+1)}$, stoga možemo primijeniti lančano pravilo

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial a_j^{(l+1)}} \cdot \frac{\partial a_j^{(l+1)}}{\partial w_{ji}^{(l)}}. \tag{1}$$

Prisjetimo se kako su aktivacijske funkcije oblika

$$a_j^{(l+1)} = \sum_i w_{ji}^{(l)} z_i^{(l)},$$

stoga su parcijalne derivacije funkcija $a_j^{(l+1)}$ s obzirom na težine $w_{ji}^{(l)}$ zadane s

$$\frac{\partial a_j^{(l+1)}}{\partial w_{ji}^{(l)}} = z_i^{(l)}.$$

Ako uvedemo oznaku $\delta_j^{(l)} := \frac{\partial E_n}{\partial a_j^{(l+1)}}$, izraz (1) možemo zapisati kao

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l)}. \tag{2}$$

Sada je potrebno izračunati $\delta_j^{(l)}$ za $l = 1, ..., L$, gdje je L ukupan broj slojeva u neuronskoj mreži.
Za $l = L$, odnosno za vrijednosti u izlaznom sloju može se pokazati kako vrijedi

$$\delta_k^{(L)} = \frac{\partial E_n}{\partial a_j^{(L+1)}} = y_k - t_k. \tag{3}$$

Općenito, za $\delta_j^{(l)}$ koji pripada nekom od skrivenih slojeva ponovnom primjenom lančanog pravila dobije se

$$\delta_j^{(l)} = \frac{\partial E_n}{\partial a_j^{(l+1)}} = \sum_k \frac{\partial E_n}{\partial a_k^{(l+2)}} \frac{\partial a_k^{(l+2)}}{\partial a_j^{(l+1)}}. \tag{4}$$

Posljednji izraz u prethodnoj jednakosti direktno slijedi iz propagacije unaprijed. Budući da je općenito za aktivaciju u sloju l vrijedi

$$a_k^{(l)} = \sum_j w_{kj}^{(l-1)} z_j^{(l-1)} = \sum_i w_{ki}^{(l-1)} h(a_i^{(l-1)}),$$

iz toga slijedi

$$\frac{\partial a_k^{(l+2)}}{\partial a_j^{(l+1)}} = w_{kj}^{(l+1)} h'(a_j^{(l+1)}),$$

uvrštavanjem u (4) dobije se

$$\delta_j^{(l)} = h'(a_j^{(l+1)}) \sum_k w_{kj}^{(l+1)} \delta_k^{(l+1)}. \tag{5}$$

Proces evaluacije funkcije pogreške započinje tako da se ulazni vektor $\mathbf{x}$ provede kroz neuronsku mrežu koristeći navedene formule za aktivacijske jedinice. Nadalje, izračunaju se vrijednosti $\delta_k^{(L)}$ koristeći formulu (4), nakon čega se unatrag računaju vrijednosti $\delta_k^{(l)}$, $l = L-1, ..., 1$ koristeći formulu (5). U posljednjem koraku, izračunate vrijednosti uvrstimo u (2) kako bi se dobila parcijalna derivacija funkcije pogreške.

2.1.3 Regularizacije u neuronskoj mreži

Prilikom procesa treniranja modela dubokog učenja, s jedne strane je cilj postići što veću točnost, ali s druge se strane treba izbjeći prenaučenost modela. Prenaučeni modeli postižu loše rezultate na novim podacima koji se u nekoj mjeri razlikuju od podataka iz skupa za treniranje. Strategija kojom se rješava ovaj problem naziva se regularizacija.
Jedan od načina regularizacije je modifikacija funkcije pogreške tako da se doda regularizirajući izraz u ovisnosti o težinama $\Omega(w)$. Tako modificirana funkcija pogreške je oblika

$$\hat{E}(w) = E(w) + \lambda\Omega(w),$$

gdje je λ parametar koji kontrolira doprinos regularizirajućeg izraza. Česti izbor za $\Omega(w)$ je norma vektora težina pa onda ovisno o izboru norme postoje različite regularizacije. Često su korištene L_1 i L_2 regularizacije.
Dodavanjem L_2 regularizacije funkcija pogreške ima sljedeći oblik

$$\hat{E}(w) = E(w) + \frac{\lambda}{2} ||w||_2^2.$$

U osnovi, ova regularizacija djeluje na vektor težina tako da one komponente koje nemaju veliki utjecaj na funkciju pogreške približi nuli.
Funkcija pogreške s L_1 regularizacijom ima oblik

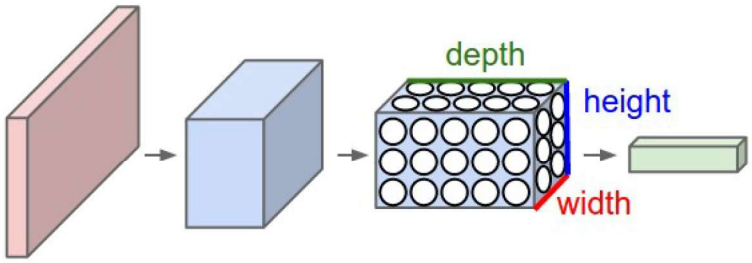
$$\hat{E}(w) = E(w) + \lambda||w||_1$$

Sljedeći način regularizacije je modifikacijom samih podataka za učenje. Činjenica je kako više podataka doprinosi učenju na način da se povećava točnost i općenitost modela. No kako je često broj dostupnih podataka ograničen, rješenje je generirati nove podatke primjenom nekih transformacija nad dostupnim podacima. Ovaj postupak je primjenjiv u klasifikaciji slika gdje možemo generirati nove slike rotirajući, skalirajući i translatirajući stare slike.

Rano zaustavljanje je način regularizacije u kojem se u svakoj iteraciji prate vrijednosti funkcije pogreške na dodatnom skupu koji se razlikuje od skupa za učenje. Tijekom procesa učenja funkcija pogreške pada kako se težine modela prilagođavaju podacima za treniranje, no računanje funkcije pogreške na dodatnom skupu podataka pokazuje da ona počinje rasti zbog prenaučenosti. Ideja je zaustaviti proces učenja, odnosno zapamtiti težine koje odgovaraju iteraciji u kojoj je vrijednost funkcije pogreške na validacijskom skupu bila najmanja. Iako se ovaj način regularizacije razlikuje od prethodno navedenih, može se detaljnije objasniti mehanizam koji stoji iza njega te pokazati kako u posebnim slučajevima postoji matematička ekvivalencije između ranog zaustavljanja i L_2 regularizacije, detaljnije u [5].

2.2 Konvolucijske mreže

U ovom poglavlju će biti detaljnije objašnjen pojam konvolucijskih mreža koje su vrlo slične neuronskim mrežama iz prethodnog poglavlja. Kao i kod neuronskih mreža, model konvolucijske mreže se sastoji od ulaznih jedinica, niza skrivenih slojeva koji sadrže aktivacijske jedinice i izlaznih jedinica. Razlika se pojavljuje u povezanosti između slojeva, a nastala je kao posljedica pretpostavke da su ulazni podaci slike. Naime, slike se prirodno reprezentiraju kao trodimenzionalne matrice, gdje dimenzije odgovaraju širini, visini i dubini koja predstavlja kanale boja, dok su komponente te matrice intenziteti boja za svaki piksel. Dakle, ulazni sloj konvolucijske mreže je trodimenzionalna matrica. Dalje u radu ćemo koristiti izraz tenzor.
Konvolucijska mreža se može podijeliti na dva dijela. Prvi dio je konvolucijska baza koja sadrži konvolucijske slojeve, slojeve s funkcijama sažimanja (engl. pooling) i aktivacijske funkcije. U ovom dijelu se izvlače generička svojstva sa slike. Drugi dio je klasifikator gdje su slojevi isti kao kod neuronskih mreža. Tu su aktivacijske jedinica potpuno povezane, primjenjuju se standardne aktivacijske funkcije i posljednji sloj rezultira vjerojatnostima pripadanja klasi.



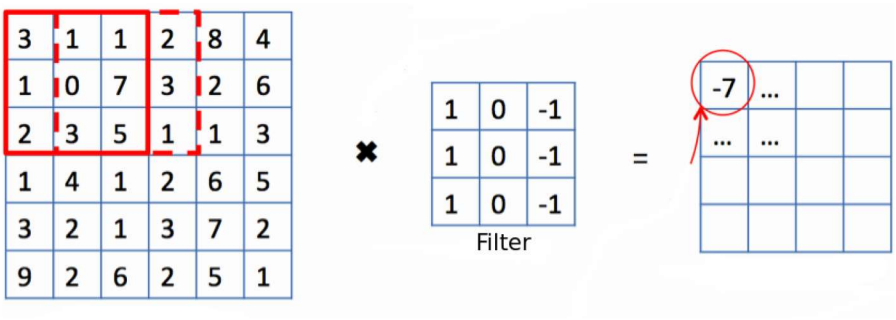
Slika 3: Grafički prikaz transformacije dimenzija volumena prilikom prolaska kroz konvolucijsku mrežu

Osnovni koncept svakog sloja je da ima ulazni tenzor nad kojim se izvrši računanje i kao rezultat se dobije tenzor koji prelazi u sljedeći sloj. Detaljnije ćemo opisati povezanost slojeva, proces računanja i dimenzije tenzora.
Svaki konvolucijski sloj sadrži niz filtera. Filteri su tenzori čija dubina odgovara dubini sloja, a širina i visina su hiperparametri koji ovise o arhitekturi. Filter se spacijalno pomiče po slici i računa se skalarni umnožak. Kao rezultat se dobije dvodimenzionalna aktivacijska mapa čija vrijednost na nekoj poziciji daje informaciju kako je filter reagirao na odgovarajuću spacijalnu poziciju slike. Izlazni volumen konvolucijskog sloja je onda niz aktivacijskih mapa od svakog filtera. Dimenzija izlaznog sloja ovisi o hiperparametrima. Dubina dakle, ovisi o broju filtera jer je dobivena slaganjem aktivacijskih mapa od svakog filtera. Širina i visina ovise o sljedećim hiperparametrima

- Korak definira za koliko mjesta će se filter pomicati po volumenu. Npr. kada je korak 1, filter se pomiče po jedan piksel.
- Uobičajeni postupak u konvolucijskom sloju je povećanje visine i širine volumena dodavanjem nul-redaka i nul-stupaca. Hiperparametar koji definira broj dodanih stupaca i redaka se naziva nadopuna (*engl.* zero-padding)

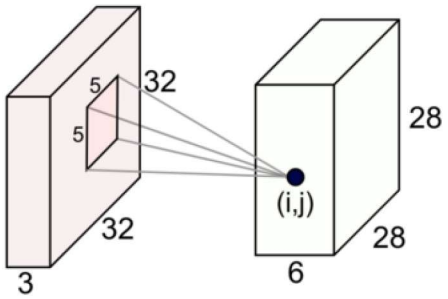
Formula za računanje spacijalne veličine izlaznog volumena u ovisnosti o W- spacijalna veličina ulaznog volumena, F- veličina filtera, S- korak, P- nadopuna glasi

$$\frac{W - F + 2P}{S} + 1.$$



Slika 4: Grafički primjer djelovanja filtera veličine $(3, 3, 1)$ na ulazni tenzor veličine $(6, 6, 1)$.

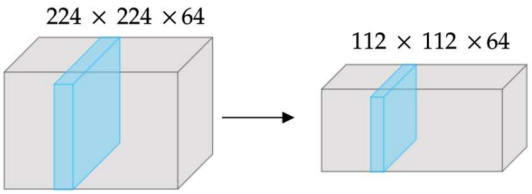
Povezanost između slojeva nije potpuna kao kod neuronskih mreža. Svaka aktivacijska jedinica u konvolucijskom sloju na nekoj spacijalnoj pozicija je povezana samo s malom regijom prethodnog sloja, ali s cijelom pripadnom dubinom. Veličina regije s kojom će se povezivati se određuje s veličinom filtera.
Navest ćemo jednostavni primjer kako bi povezali sve navedeno. Neka je dana slike veličine $32 \times 32 \times 3$. Konvolucijski sloj ima sljedeću arhitekturu 6 filtera veličine 5, korak 1 i bez nadopune. S obzirom na zadanu veličinu ulaznog volumena i hiperparametre izračunajmo veličinu izlaznog volumena. Dubina je jednaka broju filtera, dakle 6. Visina i širina se računaju prema formuli $\frac{W-F+2P}{S} + 1 = \frac{32-5+2\cdot 0}{1} + 1 = 28$.
Na spacijalnoj poziciji (i, j) u izlaznom volumenu je 6 aktivacijskih jedinica. Sve odgovaraju jednoj regiji slike veličine $5 \times 5 \times 3$, a svaka od njih pripada jednom filteru. Tako su dva sloja povezana.



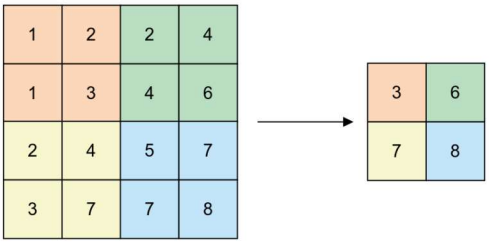
Slika 5: Aktivacijska jedinica na poziciji (i, j) je povezana s označenom regijom u prethodnom sloju.

Kako je broj parametara i dalje velik, unatoč lokalnoj, a ne potpunoj povezanosti, uvodi se koncept dijeljenja parametara. Ideja je da aktivacijske jedinice na istoj dubini u volumenu koriste iste parametre. Ovaj pristup uvelike smanjuje broj parametara.
Ako se vratimo na prethodni primjer, volumen u drugom sloju je veličine $28 \times 28 \times 6$, stoga ima $28 \cdot 28 \cdot 6 = 4704$ aktivacijskih jedinica. Svaka aktivacijska jedinica je povezana s jednom

regijom veličine $5 \times 5 \times 3$ što ukupno povlači $28 \cdot 28 \cdot 6 \cdot 5 \cdot 5 \cdot 3 = 352800$ parametara. Ako se uvede dijeljenje parametara, tada svih $28 \cdot 28$ aktivacijskih jedinica koriste iste parametre pa je ukupan broj $6 \cdot 5 \cdot 5 \cdot 3 = 450$.
Kao što je već navedeno, osim konvolucijskih slojeva postoje i slojevi sažimanja. Njihova uloga je smanjivanje spacijalne veličine volumena. Za zadani volumen, operacije se izvršavaju za svaku dubinu posebno. Najčešće operacije koje se primjenjuju nad dvodimenzionalnim slojem volumena su računanje maksimuma i prosjeka. Analogno kao i u konvolucijskom sloju, postoji filter za kojeg se odabiru hiperparametri veličine i korak. Filter se pomiče po ulaznom volumenu na svakoj dubini i računa se prosjek ili maksimum. Formula za računanje veličine izlaznog volumena je ista kao za konvolucijski sloj, no bez nadopune $\frac{W-F}{S} + 1$.



(a) Grafički prikaz djelovanja funkcije sažimanja. Spacijalna veličina slike se smanjuje prema navedenoj formuli, dok dubina ostaje ista.



(b) Primjer funkcije sažimanja maksimumom

Slika 6: Sloj s funkcijom sažimanja (*engl. pooling*)

(a) Prije djelovanja funkcijom sažimanja.

(b) Nakon djelovanja funkcijom sažimanja.

Slika 7: Primjer djelovanja funkcije sažimanja maksimumom na sliku s magnetske rezonance koljena. Slika je dimenzije 224×224, nakon djelovanja funkcijom sažimanja dobije se dimenzija 112×112.

2.2.1 Poznate arhitekture konvolucijskih mreža i učenje prenošenjem

Postoji nekoliko poznatih modela konvolucijskih mreža čije je važnost u poboljšavanju dotadašnje efikasnosti i točnosti i tome što su dostupne za korištenje prilikom treniranja vlastitih modela. Neka od popularnijih imena konvolucijskih mreža su GoogLeNet, AlexNet,

ResNet, VGGNet. Arhitektura AlexNet-a je korištena u praktičnom dijelu, stoga će biti detaljnije opisana.
AlexNet je treniran i testiran na skupu podataka pod nazivom ImageNet koji sadrži preko 15 milijuna označenih slika.

Opisat ćemo ukratko propagaciju kroz slojeve AlexNet-a.

1. Slike na ulaznom sloju su veličine $(224, 224, 3)$. U prvom konvolucijskom sloju se koristi 96 filtera, veličine $11 \times 11 \times 3$, s korakom 4 iz čega slijedi da je veličina izlaznog sloja $55 \times 55 \times 96$.

2. Izvršava se funkcija sažimanja i aktivacijska funkcija (ReLU)

3. Sljedeći konvolucijski sloj je veličine $5 \times 5 \times 48$, koristi 256 filtera veličine $5 \times 5 \times 48$ pa je veličina izlaznog volumena $27 \times 27 \times 256$.

4. Ponovno se primjenjuje sažimanje i aktivacijske funkcije (ReLU).

5. U trećem konvolucijskom sloju se koristi 384 filtera veličine $3 \times 3 \times 256$) i rezultat je volumen veličine $13 \times 13 \times 384$.

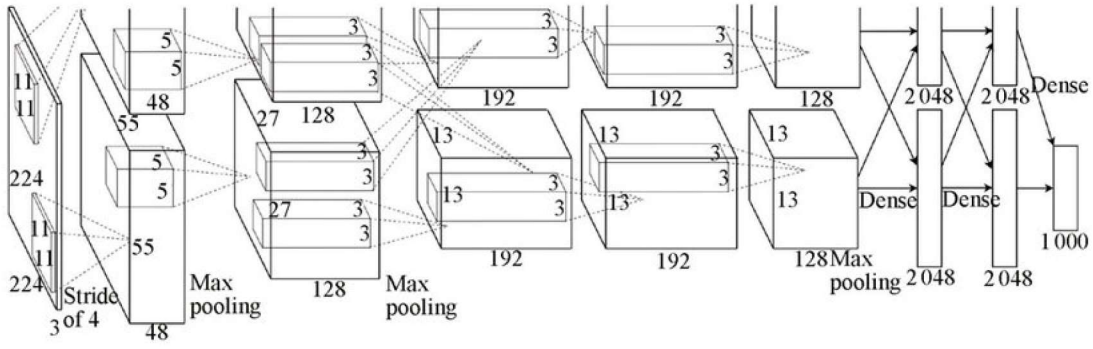
6. Četvrti sloj koristi 384 filtera veličine $3 \times 3 \times 192$ i rezultira volumenom $13 \times 13 \times 256$.

7. Peti sloj koristi 256 filtera veličine $3 \times 3 \times 192$. Izlazni volumen je $13 \times 13 \times 256$

8. Izvršava se funkcija globalnog sažimanja maksimumom kako bi se smanjio na $6 \times 6 \times 256$.

9. Sljedeća dva sloja su potpuno povezana, oba imaju 4096 aktivacijskih jedinica.

10. Izlazni sloj sadrži 1000 aktivacijskih jedinica, odnosno vjerojatnosti pripadanja jednoj od klasa.



Slika 8: Ilustracija arhitekture AlexNeta.

Odabir arhitekture i veličina dostupnog skupa podataka utječu na rezultate konvolucijske mreže. U praksi se rijetko konstruira i trenira konvolucijsku mrežu od početka. Razlog tome su nedostatak dovoljno velikog skupa podataka i vremenska zahtjevnost. Rješenje je primjena učenja prenošenjem. Umjesto izvršavanja procesa učenja od početka koriste se postojeći modeli uz dodatna prilagođavanja.
Prva strategija je uklanjanje zadnjeg, klasifikacijskog sloja pretrenirane mreže, dok ostatak arhitekture ostaje nepromijenjen i koristi se za izvlačenje značajki sa slike. Nakon što se

izvrši cijeli proces računanja za sve podatke, trenira se linearni klasifikator na dobivenim izlaznim vrijednostima. Ovaj proces se naziva izvlačenje značajki (*engl.* feature extraction). Sljedeća strategija uključuje podešavanje težina i na prethodnim slojevima. Kako prvi slojevi odgovaraju prepoznavanju nekih generičkih značajki, oni se obično ostave netaknutim, dok se parametri na posljednjim slojevima podešavaju jer odgovaraju nekim značajkama specifičnim za problem klasifikacije koji se rješava.
U nekim slučajevima je izbor treniranje modela od početka koristeći zadanu arhitekturu.
Izbor metode za učenje prenošenjem se odlučuje na temelju veličine dostupnog skupa podataka za treniranje i sličnosti problema s postojećim pretreniranim modelima.

- Za veliki skup podataka koji se razlikuje od skupa podataka pretreniranog modela je najbolja trenirati postojeći model od početka
- Za veliki skup podataka koji je sličan skupu podataka pretreniranog modela je najbolje odabrati fine-tuning, odnosno podešavanje parametara konvolucijskih i klasifikacijskih slojeva.
- Za mali skup podataka koji se razlikuje od skupa podataka pretreniranog modela je najbolje trenirati klasifikator počevši od ranijih slojeva.
- Za mali skup podataka koji je sličan skupu podataka pretreniranog modela je najbolje trenirati linearni klasifikator na posljednjem sloju.

2.3 Optimizacijski algoritmi

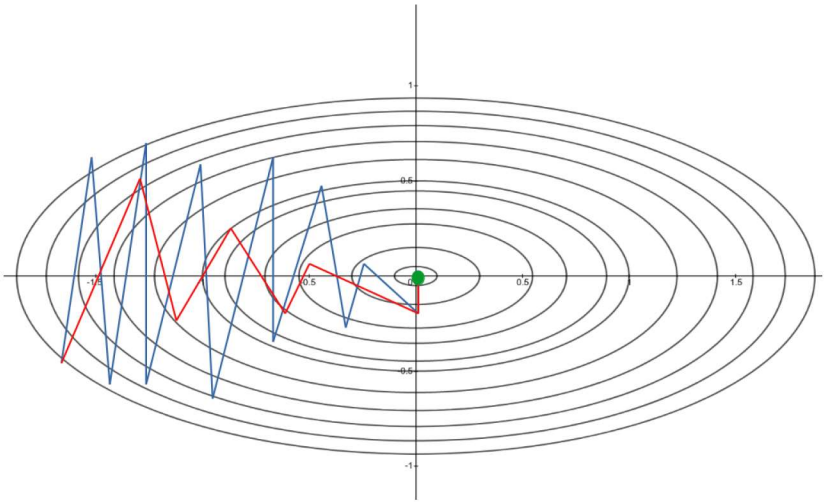
Optimizacijski algoritmi su osnova procesa strojnog učenja, no za razliku od uobičajene optimizacije gdje je cilj pronaći optimalne parametre, ovdje je cilj postići što veću točnost modela. U poglavlju 2.1.2 smo se susreli s gradijentnom metodom i vidjeli kako se može izračunati gradijent funkcije pogreške. U ovom poglavlju ćemo se nadovezati na to i navesti nekoliko novih pristupa optimizaciji parametara.
Nedostatak gradijentne metode je evaluacija funkcije na cijelom skupu podataka za treniranje koji treba biti što veći kako bismo postigli što bolje rezultate. Stoga se problemu može pristupiti tako da se gradijent funkcije pogreške računa za svaki ulazni podatak što se naziva stohastički gradijentni spust (engl. Stochastic Gradient Descent - SGD). U praksi se najčešće koristi pristup u kojem se gradijent funkcije evaluira na manjem podskupu podataka tzv. *mini-batch* metode. Iz cijelog skupa podataka veličine m se slučajnim odabirom uzima grupa (*engl.* batch) podataka veličine m' na temelju koje se procjenjuje gradijent.

Kako stohastički gradijentni spust može sporo konvergirati, učenje se može ubrzati metodama koje koriste moment. U osnovi algoritma je uvođenje nove varijable v koja sadrži informaciju o brzini i smjeru kretanja parametara. Postupak se svodi na akumuliranje prosjeka gradijenta u varijabli v i ažuriranje težina u smjeru vektora v

$$v = \mu v - \alpha \nabla E(w)$$

$$w_{k+1} = w_k + v$$

Hiperparametar α je stopa učenja, a μ određuje utjecaj prethodno izračunatih gradijenata na smjer ažuriranja.
Za razliku od običnog gradijentnog spusta, ovdje se težine ažuriraju izračunatim prosjekom, odnosno u obzir uzimamo kako se gradijent mijenja kroz iteracije. Ako se gradijent u određenom smjeru izrazito smanjuje, onda u tom smjeru želimo napraviti veći korak. Na



Slika 9: Ilustrativni prikaz koraka u optimizaciji s običnim gradijentnim spustom (plava boja) i s momentom (crvena boja).

slici 9 je ilustrativno prikazana ova ideja. Rezultat takvog računanja težina je manji broj koraka potrebnih za postizanje optimalnog rješenja.

Nesterov moment je modifikacija navedenog algoritma. Razlikuju se u evaluaciji gradijenta. Glavna ideja iza Nesterovog momenta je izračunati vrijednost gradijenta u sljedećoj aproksimaciji, zatim izračunati prosječni gradijent i korigirati parametre

$$v = \mu v - \alpha \nabla E(w + \mu v)$$

$$w = w + v.$$

Optimizacija parametara neuronske mreže se može rješavati primjenjujući algoritme koji koriste druge parcijalne derivacije za ažuriranje. Osnovni takav algoritam je Newtonova metoda koja se računa sljedećim iterativnim postupkom:

$$w_{k+1} = w_k + s_k,$$

gdje je s_k rješenje jednadžbe

$$\nabla^2 f(w_k)s = -\nabla f(w_k).$$

Možemo primijetiti kako ovaj postupak ne sadrži dodatne hiperparametre. Nedostatak ove metode je što je računanje drugih parcijalnih derivacija, odnosno Hessijana funkcije zahtjevno. U praksi se koriste Kvazi-Newtonove metode.

Općenito u neuronskim mrežama, osim ažuriranja težina na spojevima aktivacijski jedinica, potrebno je optimizirati i ostale hiperparametre koji se pojavljuju. Stopa učenja je hiperparametar koji je najteže postaviti jer njegova vrijednost značajno utječe na model. U nastavku će biti navedeno nekoliko algoritama koji su najčešće upotrebljavani.

AdaGrad algoritam adaptira stopu učenja tako da ju skalira s obzirom na akumulirane vrijednosti gradijenta pripadne funkcije. Tako će parametri koji imaju velike parcijalne derivacije brzo početi koristiti male stope učenja, dok parametri s malim parcijalnim derivacijama neće imati velike promjene u stopi učenja.

Algorithm 1: AdaGrad algoritam

Izabrati: Globalnu stopu učenja α, početnu vrijednost parametra θ, konstantu δ koja obično iznosi 10^{-7}

Inicijalizirati varijablu $r = 0$ koja akumulira gradijent.

while *zaustavni kriterij nije ispunjen* **do**

uzeti uzorak od m podataka iz trening skupa $x_1, ..., x_m$ zajedno s pripadnim oznakama $y_1, ...y_m$

Izračunati gradijent: $\mathbf{g} \leftarrow \frac{1}{m} \nabla \sum_i E_i(\theta)$

Akumulirati kvadratni gradijent: $r \rightarrow r + g \cdot g$

Izračunati ažuriranu vrijednost: $\Delta\theta \rightarrow -\frac{\alpha}{\delta + \sqrt{r}} \cdot g$

Primjeniti ažuriranu vrijednost: $\theta \rightarrow \theta + \Delta\theta$

end

RMSProp algoritam je modifikacija AdaGrad algoritma za bolju optimizaciju nekonveksnih funkcija. RMSProp algoritam akumulira vrijednosti gradijenta tako da prigušuje stare vrijednosti gradijenta.

Algorithm 2: RMSProp algoritam

Izabrati: Globalnu stopu učenja α, početnu vrijednost parametra θ, konstantu δ koja obično iznosi 10^{-6}, stopu opadanja ρ

Inicijalizirati varijablu $r = 0$ koja akumulira gradijent.

while *zaustavni kriterij nije ispunjen* **do**

uzeti uzorak od m podataka iz trening skupa $x_1, ..., x_m$ zajedno s pripadnim oznakama $y_1, ...y_m$

Izračunati gradijent: $\mathbf{g} \leftarrow \frac{1}{m} \nabla \sum_i E_i(\theta)$

Akumulirati kvadratni gradijent: $r \rightarrow \rho r + (1 - \rho) g \cdot g$

Izračunati ažuriranu vrijednost: $\Delta\theta \rightarrow -\frac{\alpha}{\delta + \sqrt{r}} \cdot g$

Primjeniti ažuriranu vrijednost: $\theta \rightarrow \theta + \Delta\theta$

end

Optimizacijski algoritam **Adam** čiji je naziv izveden iz *adaptive moment adaptation*, predstavljen je u radu [6]. Algoritam je dizajniran tako da kombinira svojstva navedenih algoritama AdaGrad i RMsProp. Za zadanu funkcija cilja f, koja ovisi o parametrima θ računa se gradijent, $\nabla f(\theta)$ tako da se u svakom koraku aproksimiraju momenti prvog i drugog reda.

Algorithm 3: Adam algoritam

Izabrati: Globalnu stopu učenja α, parametre $\beta_1, \beta_2 \in [0, 1)$, stohastička funkcija cilja f u ovisnosti o parametrima θ, početna aproksimacija θ_0

Inicijalizirati: $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$

while *zaustavni kriterij nije ispunjen* **do**

$t \leftarrow t + 1$

Izračunati gradijent: $g_t \leftarrow \nabla f_t(\theta_{t-1})$

Ažurirati vrijednost prvog momenta: $m_t \leftarrow \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t$

Ažurirati vrijednost drugog momenta: $v_t \leftarrow \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2$

Korekcija prvog momenta: $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ Korekcija drugog momenta: $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$

Ažurirati parametar: $\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$

end

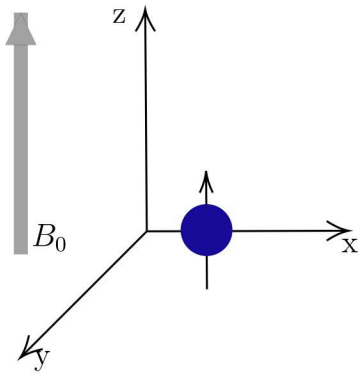
3. Praktični projekt

U praktičnom projektu se koriste konvolucijske mreže za klasifikaciju slika s magnetske rezonance. Problem klasifikacije na takvom skupu podataka je prepoznavanje zdravog i oštećenog tkiva, odnosno prepoznavanje različitih vrsta oštećenja.
Standardno dijagnosticiranje na temelju slike s magnetske rezonance je proces koji je vremenski zahtjevan i potencijalno podložan pogreškama, stoga ovdje nastaje motivacija za automatizacijom istog. Postupak razvijanja takvog automatiziranog algoritma obuhvaća skupljanje i obradu velikog broja snimki magnetske rezonance, dijagnosticiranje od strane stručnog osoblja i odabir i treniranje odgovarajuće neuronske mreže.
U ovom radu će se klasificirati slike s magnetske rezonance koljena. Upravo je koljeno najčešće snimano magnetskom rezonancom, a neke tipične ozljede koje se pojavljuju su ozljede prednjeg križnog ligamenta i ozljede meniskusa.
U nastavku će biti ukratko objašnjen postupak magnetske rezonance, zatim će biti predstavljene tehnike pomoću kojih se implementiraju konvolucijske mreže. Budući da postoje znanstveni radovi koji se bave navedenim problemom koji su poslužili kao motivacija za implementiranjem vlastitog rješenja, u nastavku će biti dan njihov pregled i postignuti rezultati.

3.1 Magnetska rezonanca

Magnetska rezonanca [1] je tehnika za izradu slojevitih slika unutrašnjosti ljudskog tijela. Zbog mogućnosti detaljnog prikaza različitih tkiva ima vrlo čestu i široku upotrebu. Koristi se kod dijagnosticiranja neuroloških bolesti, bolesti jetre, otkrivanja tumora i cista u mekim tkivima, ozljeda i bolesti zglobova.
Tehnika magnetne rezonance zasniva se na promatranju jezgri atoma i njihovom ponašanju u magnetskom polju. Konkretnije, promatra se atom vodika jer on zbog svojih kemijskih i fizikalnih svojstava daje najbolje rezultate pri procesu magnetske rezonance. Jezgra atoma vodika sadrži jedan proton. U nastavku, proton i njegovo ponašanje u magnetskom polju se može vizualizirati kao slabi magnet koji ima polove i os rotacije. Pretpostavimo da promatramo prostor kao Kartezijev koordinatni sustav. Magnetsko polje, B_0, kojem će biti izloženi atomi vodika položeno je u smjeru z osi, orijentirano prema gore. Protoni se smještaju u prostor tako da leže na z-osi, ali mogu biti orijentirani prema gore ili dolje, kao što se može vidjeti na slici 10. Takvo stanje se naziva prirodnim stanjem ili ekvilibrijem. U slučaju kada su protoni orijentirani kao i vanjsko magnetno polje, odnosno prema gore, kažemo da su u niskom energetskom stanju, a u suprotnom da su u visokom energetskom stanju. Kao što je već navedeno, protoni imaju os oko koje rotiraju, no osim toga ta os rotacije pravilno mijenja smjer s obzirom na z-os. Ta pojava se naziva precesija.

[1] MRI magnetic resonance imaging

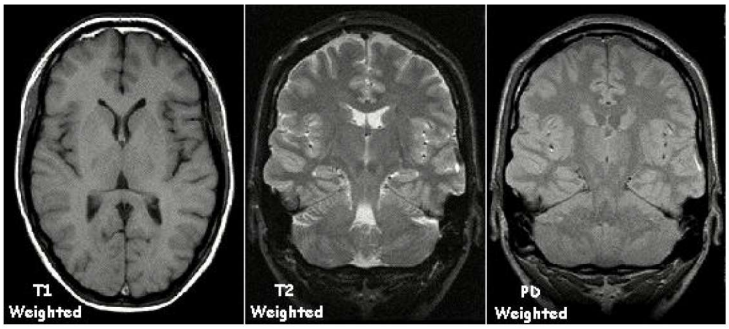


Slika 10: Koordinatni sustav pomoću kojeg se opisuje ponašanje magnetskog polja.

Kada bi na protone djelovala dodatna vanjska sila, pomaknuli bi se iz prirodnog položaja pri čemu bi nastavili precesirati. Upravo je to ono što se događa u procesu magnetske rezonance. Emitira se signal takav da protone pomakne u transverzalnu ravninu i da mu je frekvencija jednaka frekvenciji precesirajućeg protona. Poklapanjem dvije frekvencije događa se rezonanca.
Uređaj za magnetsku rezonancu sadrži zavojnicu koja emitira taj signal, ali isto tako prima signal od čestica koje precesiraju. Signalom se može upravljati tako da se varira njegovo trajanje i broj ponavljanja te se dobiju drugačiji rezultati. Ovisno o duljini trajanja izlaganja čestica vanjskom signalu, može se postići pomicanje protona iz prirodnog položaja za željeni kut. Uobičajeno je prvo primijeniti signal koji će protone pomaknuti za 90 stupnjeva od prirodnog položaja, nakon toga se može primijeniti dvostruko dulje kako bi ih se odmaklo za 180 stupnjeva. Moguće je i višestruko, uzastopno primjenjivanje signala. Signal se može izračunati i kao zbroj doprinosa svake atomske jezgre. Raspored jezgri se može promatrati jednodimenzionalno tako da su poslagane duž jedne osi ili dvodimenzionalno. Za tako raspoređene jezgre, promatra se njihova distribuiranost, odnosno funkcija gustoće u ovisnosti o poziciji. Primjenom Fourierovih transformacija na signal izračuna se funkcija gustoće koju se koristi u konstruiranju slike. Ovakva tehnika se naziva Proton density (PD) tehnika.

Sljedeće dvije tehnike se vežu uz pojmove T_1 i T_2 relaksacije. T_1 relaksacija je pojava da se proton vrati u svoje početno stanje. To stanje se naziva niskim energetskim stanje, što povlači da proton prelazi iz visokog energetskog stanja i pri tome se oslobađa količina energije u okolinu. T_1 relaksacija je eksponencijalni proces. T_2 relaksacija se odnosi na progresivno izlaženje rotirajućih dipola iz faze. To se događa zbog svojstava samih atoma i okolnog tkiva. Naime, rotirajući protoni tvore oko sebe mala magnetska polja unutar kojih, kad se djeluje s vanjskom silom, događa se izmjena energije između protona. Brže kretanje uzrokuje dužu T_2 relaksaciju.

Kada se promatra T_1 relaksaciju, protoni u masnim tkivima se brzo vraćaju u početno stanje. Kao rezultat, masna tkiva su na slikama u svijetlim nijansama. S druge strane, molekule vode imaju dužu T_1 relaksaciju i na slikama su u tamnim nijansama.
U T_2 tehnici, tetive i čvrsta tkiva se pojavljuju u tamnim nijansama, masna tkiva u srednje svijetlim nijansama i voda u najsvjetlijim nijansama. Kod PD tehnike, tkiva s većom gustoćom protona imaju svjetlije nijanse. Primjere ovih tehnika možemo vidjeti na Slici 11



Slika 11: Primjeri T_1, T_2 i PD snimki magnetne rezonance.

Na početku su uvedene neke pretpostavke o položaju magnetskog polja i cijeli proces je opisan s obzirom na definirani koordinatni sustav. Ovisno o potrebama, koordinatni sustav se može položiti u prostor tako da se izvodi magnetska rezonanca tijela u različitim ravninama. Osnovne anatomske ravnine su: sagitalna koja dijeli ljudsko tijelo na lijevu i desnu polovinu, koronalna koja dijeli ljudsko tijelo na prednji i stražnji dio i transverzalna ravnina koja dijeli ljudsko tijelo na gornji i donji dio. Na Slici 3 su prikazane snimke magnetne rezonance koljena snimane u tri navedene ravnine.

Osim što su snimki različitih ravnina, često je potrebno snimanje u 3 dimenzije. Jedan od načina kako se to može izvesti je uzastopnom primjenom dvodimenzionalnih snimanja tako da se pulsevi višestruko emitiraju, različitim frekvencijama kako bi pogodili određeni sloj tkiva.

3.2 PyTorch

U ovom poglavlju će ukratko biti argumentirano korištenje programskog paketa Pytorch. Pytorch je baziran na programskom jeziku Python. Primjenjuje se u metodama dubokog učenja jer koristi mogućnosti grafičke procesorske jedinice (GPU) i pruža veliku fleksibilnost i brzinu.
Proces treniranja modela dubokog učenja je vremenski zahtjevan i njegovo izvođenje na procesoru - CPU dugo traje, stoga je rješenje prebacivanje računanja na grafički procesor. Grafička procesorska jedinica sadrži velik broj jezgri koje su optimizirane za paralelno računanje. U PyTorchu je dostupan paket koji podržava izvršavanje CUDA operacija.

U PyTorch paketu je implementiran veliki broj gotovih funkcija za rad s neuronskim mrežama. Osnovni objekt nad kojim se izvršavaju operacije je tenzor. Tenzori se direktno mogu premještati s CPU na GPU i obratno. Navest ćemo nekoliko paketa specijaliziranih za strojno učenje.
PyTorch sadrži paket *nn* koji omogućuje jednostavno slaganje konvolucijskih mreža, automatizira procese računanja i sadrži implementacije velikog broja funkcija pogreške.
Koristeći sljedeće naredbe slaže se neuronska mreža tako da se navedu slojevi koji se sekvencijalno izvršavaju na zadanom ulazu. Pokazat ćemo na primjeru jednostavne mreže koja sadrži dva linearna sloja (Kod 1). Kreiramo tenzore zadanih veličina sa slučajno generiranim vrijednostima za ulazne podatke x i njima pripadne izlazne podatke y naredbom *nn.Sequential* povežemo slojeve.

```
1 N, n_in, n_hidden, n_out = 50, 1000, 100, 10
2 x = torch.randn(N, n_in)
```

```
y = torch.randn(N, n_out)
model = torch.nn.Sequential(
        torch.nn.Linear(n_in, n_hidden),
        torch.nn.Linear(n_hidden, n_out))
```

Kod 1: Konstruiranje neuronske mreže koristeći nn.Sequential

U slučaju kada je potrebno konstruirati kompleksniji model kreira se klasa koja nasljeđuje klasu Module tako da se željeni slojevi instanciraju kao atributi klase i definira se propagacija unaprijed u metodi *forward*.

```
class MyNet(torch.nn.Module):
    def __init__(self, n_in, n_hidden, n_out)
        super(MyNet, self).__init__()
        self.linear1 = torch.nn.Linear(n_in, n_hidden)
        self.linear2 = torch.nn.Linear(n_hidden, n_out)

    def forward(self, x):
        x = self.linear1(x)
        x = self.linear2(x)
        return x
```

Kod 2: Konstruiranje neuronske mreže koristeći nn.Module

PyTorch paket *optim* pruža implementaciju velikog broja optimizacijskih algoritama poput Adam, Adagrad, LBFGS itd. U navedenom primjeru koristimo model konvolucijske mreže iz prethodnog primjera, za optimizator odabiremo Adam s proizvoljno odabranom stopom učenja. Proces treniranja ponavljamo 1000 puta tako da izvršimo propagaciju unaprijed, izračunate parametre spremimo u varijablu *_pred*. Vrijednost funkcije pogreške se računa koristeći *BinaryCrossEntropy* funkciju i spremaju u varijablu *loss*. Prije računanja gradijenta funkcije pogreške s obzirom na težine modela, svi gradijenti se postavljaju na nulu. Razlog tome je što se u svakoj iteraciji vrijednosti gradijenata ne prepisuju nego akumuliraju. Na kraju pozivajući funkciju *step* ažuriramo vrijednosti težina.

```
model = MyNet()
loss_fn = torch.nn.BinaryCrossEntropy()

learning_rate = 1e-4
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

for step in range(1000):
  _pred = model(x)

    loss = loss_fn(y_pred, y)

    optimizer.zero_grad()

    loss.backward()

    optimizer.step()
```

Kod 3: Primjer implementacije procesa treniranja

PyTorch pruža različite alate za pripremu i učitavanje podataka. torch.utils.data.Dataset je apstraktna klasa koja predstavlja skup podataka. Potrebno je implementirati vlastiti klasu koja nasljeđuje klasu Dataset i prepisati metode __len__ i __getitem__. Koristeći ovu klasu možemo jednostavno pristupiti podacima i učitati ih na proizvoljan način. Osim toga, na podatke se mogu primijeniti transformacije za promjenu oblika i veličine.

Paket torchvision sadrži modele trenirane na ImageNet skupu slika. Dostupni su ResNet, AlexNet, VGG, SqueezeNet, DenseNet i Inception. Modeli se mogu koristiti u cijelosti sa svim pretreniranim težinama ili se može proizvoljni broj slojeva, odnosno težina ponovno trenirati s obzirom na podatke. U sljedećem primjeru je upotrebljen AlexNet s pripadnim parametrima koristeći argument $pretrained = true$.

```
class MyNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = models.alexnet(pretrained=True)

        for param in self.model.parameters():
            param.requires_grad = False
        self.classifier = nn.Linear(256,10)

    def forward(self, x):
        x = self.model.features(x)
        x = self.classifier(x)
        return x
```

Kod 4: Primjer učenja prenošenjem koristeći pretrenirani AlexNet.

Parametri modela se spremaju u *state_dict*, Pythonov rječnik koji svakom sloju mapira pripadni tenzor s parametrima. Potrebno je naglasiti kako se spremaju samo slojevi koji imaju parametre, npr. konvolucijski sloj ili linearni sloj. Osim modela, objekti poput optimizatora imaju *state_dict* u koji se spremaju pripadni parametri. Spremanje i učitavanje parametara se radi koristeći naredbe *save* i *load*. Može se spremiti samo potrebne parametre ili čitavi model.

```
torch.save(model.state_dict(), PATH)
model = TheModelClass(*args, **kwargs)
model.load_state_dict(torch.load(PATH))
model.eval()
```

Kod 5: Spremanje i učitavanje naučenih parametara modela.

```
torch.save(model, PATH)
model = TheModelClass(*args, **kwargs)
model = torch.load(PATH)
model.eval(
```

Kod 6: Spremanje i učitavanje cijelog modela.

Spremanje i učitavanje modela je moguće u kombinaciji s korištenjem grafičke procesne jedinice na kojoj se model i parametri mogu spremati i učitavati, ali isto tako se mogu spremati na grafičkoj procesnoj jedinici i učitavati na centralnoj ili obrnuto.

3.3 Pregled postojećih radova

3.3.1 Deep-learning-assisted diagnosis for knee magnetic resonance imaging: Development and retrospective validation of MRNet

Prvi istraživački rad koji ćemo pregledati je Deep-learning-assisted diagnosis for knee magnetic resonance imaging: Development and retrospective validation of MRNet - [1] proveden na Sveučilištu Stanford. Podaci skupljeni na Stanford University Medical Centre sadrže 1370 magnetskih snimki koljena, među kojima je 1104 sadržavalo oštećenja. Ozljeda prednjeg križnog ligamenta (*engl.* anterior cruciate ligament - ACL) je sadržana na 319 uzoraka, ozljeda meniskusa na 508 uzoraka, a kombinacija obje ozljede na 194 uzorka. Snimke su napravljene standardnim tehnikama magnetske rezonance: koronalna T1 i T2, sagitalna PD (proton density) i T2 te aksijalna PD tehnika.
Podaci su podijeljeni u tri skupa: za treniranje, podešavanje i validaciju.
Dodatno je korišten skup slika s magnetske rezonance koljena iz Kliničkog bolničkog centra Rijeka. Skup sadrži 917 snimki, napravljenih sagitalnom PD tehnikom. Među uzorcima je 690 zdravih, 172 s djelomičnim oštećenjima i 55 s potpunim oštećenjima. Podaci su s obzirom na stanje koljena anotirani s oznakama 0-zdravo, 1-djelomično ozlijeđeno, 2- potpuno ozlijeđeno.
Ovdje je potrebno naglasiti kako se ulazni podaci za konvolucijsku mrežu razlikuju od klasifikacije standardnih slika. Slike se inače reprezentiraju u tri dimenzije kao $v \times š \times d$, gdje je v visina, $š$ širini i d kanal boja. U ovom slučaju ulazni podatak je niz takvih slika, stoga će imati četiri dimenzije, $s \times v \times š \times d$, gdje je s broj slika u nizu. Kao što je navedeno u poglavlju o magnetskoj rezonanci, kako bi se dobila trodimenzionalna snimka koljena napravi se niz snimki prolazeći kroz jednu od ravnina ljudskog tijela. Takav niz snimki jednog koljena ćemo nazivati volumen.

Slike su obrađene prije procesa treninga na sljedeći način: veličine su skalirane na 256×256 piksela, intenziteti su skalirani s obzirom na standardiziranu distribuciju intenziteta piksela u skupu podataka za treniranje, zatim su vrijednosti intenziteta odrezane na segment $[0, 255]$. Budući da broj podataka nije dovoljno velik za treniranje konvolucijske mreže od početka korištena je tehnika učenja prenošenjem. Kao polazište je uzet konvolucijski dio AlexNet-a. Maknuti su slojevi linearnog klasifikatora tako da je tok podataka završio s tenzorima veličine $s \times 256 \times 7 \times 7$ [2], gdje je s broj slika u volumenu. Dodan je sloj s funkcijom globalnog sažimanja prosjekom(engl. Global average pooling - GAP) koji djeluje na svaki od s tenzora tako da za svaki sloj veličine 7×7 računa prosječnu vrijednost. Izlazna veličina tenzora je tada s. Zatim je dodan sloj s funkcijom sažimanja maksimumom (engl. Global max pooling) koji vraća tenzor veličine 256. Na kraju je dodan potpuno povezani sloj za klasifikaciju sa sigmoid funkcijom kako bi dobili predikcije pripadanja nekoj od klasa. Za funkciju pogreške korištena je binary cross-entropy funkcija.
Za svakog ispitanika postoje volumeni snimani u sagitalnoj T_2 tehnici, koronalnoj T_1 tehnici i aksijalnoj PD tehnici. Dodatno su predikcije za svaki volumen s pripadnom originalnom oznakom trenirane koristeći logističku regresiji kako bi se dobila jedna, zajednička predikcija.

Kako bi se izmjerila kvaliteta modela napravljeni su brojni statistički testovi. Glavna usporedba je izvršena između rezultata dobivenih modelom i dijagnoza postavljenih od strane liječnika radiologa. Provjera kvalitete se vršila na temelju kriterija točnosti, specifičnosti i osjetljivosti. Pri dijagnosticiranju oštećenja općenito, nije bilo statističke razlike u perfor-

[2] U dokumentaciji AlexNeta posljednji konvolucijski sloj završava s tenzorom veličine $256 \times 6 \times 6$.

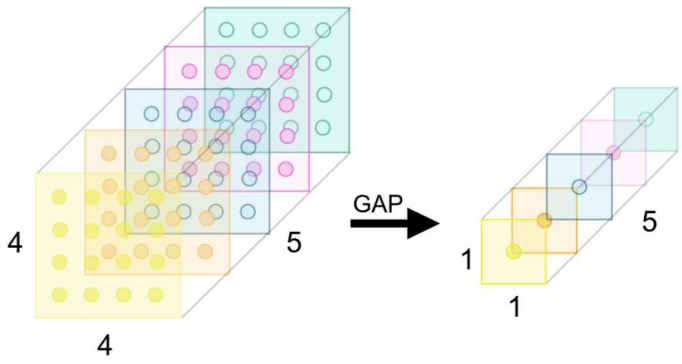
mansu modela i radiologa, iako su rezultati radiologa za sva tri kriterija bili bolji. Model je pokazao bolje rezultate za specifičnost dijagnosticiranja ozljeda prednjeg križnog ligamenta. Pri dijagnosticiranju ozljeda meniskusa, bolji rezultati su bili od strane radiologa. Općenito, razlike u performansima nisu statistički značajne. U sljedećoj statističkoj analizi su ispitali korisnost modela prilikom dijagnosticiranja ozljeda. Tako su uspoređene dijagnoze koje su liječnici napravili sami i one koje su napravljene uz pomoć modela. Razlika se pojavila kod dijagnosticiranja ozljeda prednjeg križnog ligamenta u smislu da rezultati modela imaju bolju specifičnost. Potencijalna posljedica ovoga bi mogla biti smanjivanje broja krivo potvrđenih ozljeda ligamenta.
Na kraju su analizirali rezultate dobivene djelovanjem modela na podacima iz Riječkog kliničkog bolničkog centra. Točnost od 0.824 je postignuta na validacijom bez dodatnog treniranja modela. Dok je točnost od 0.911 postignuta nakon podešavanja parametara. Ovdje se može zaključiti kako model treniran na jednom skupu podataka daje prihvatljive rezultate na slikama iz novog skupa podataka, no kako bi se postigla bolja preciznost potrebno je dodatno trenirani model. Navedene činjenice otvaraju problem stvaranja dovoljno općenito modela koji će imati visoku točnost na podacima koji dolaze iz različitih skupova. U praksi bi to značilo imati model koji daje što točnije dijagnoze na snimkama koje su napravljene različitim tehnikama, na različitim uređajima.

Kako bi provjerili da se tijekom procesa treniranja uče odgovarajući uzorci sa slika implementirali su aktivacijske mape (*engl.* class activation mapping - CAM). Nakon prolaska kroz konvolucijsku mrežu, za dobiveni rezultat reverznim se postupkom traže regije u uzlaznom volumenu koje su doprinijele tom rezultatu. Očekuje se da te regije budu one na kojima se prepoznaje dio koljena koji je od interesa. Na primjer dio slike na kojem se točno vidi oštećeni prednji križni ligament. Dakle, aktivacijske mape za određenu klasu će pokazivati regiju slike na temelju koje je konvolucijska mreža dala predikciju. Detaljnije o aktivacijskim mapa će biti navedeno kasnije u pregledu rada [10] i u poglavlju 3.3.2.
Rezultati dobiveni aktivacijskim filterima su pokazali da je u većini slučajeva lokalizacija ozljede na snimci ispravna.

Model koji su razvili je postigao visoke rezultate prilikom testiranja na podacima koji dolaze iz istog skupa, dok je pri testiranju na drugom skupu podataka imao manju točnost. Nakon ponovnog treniranja je postigao visoku točnost i na novom skupu podataka. Možemo zaključiti kako model nije dovoljno općenit.
Dijagnosticiranje ozljeda od strane liječnika uz pomoć predikcija i lokalizacije modela se pokazalo iznimno korisnim i efikasnim.

3.3.2 Aktivacijske mape

Aktivacijske mape su napravljene u radu [10] Learning Deep Features for Discriminative Learning. U radu su korištene konvolucijske mreže AlexNet, VGGNet i GoogLeNet. Predložen je sljedeći postupak za generiranje aktivacijskih mapa. Iz modela se ukloni cijeli klasifikacijski dio, na zadnji konvolucijski sloj se dodaje funkcija sažimanja prosjekom (engl. Global average pooling - GAP). GAP djeluje na volumen tako da izračuna spacijalni prosjek na svakoj dubini i te vrijednosti se onda lineariziraju koristeći neki linearni klasifikator.



Slika 12: Grafički prikaz djelovanja funkcije globalnog sažimanja prosjekom (*engl.* Global average pooling - GAP)

Formalno se gornji postupak može zapisati na sljedeći način. Neka je $f_k(x, y)$ aktivacijska jedinica na poziciji (x, y) na k-toj dubini. Tada je rezultat funkcije globalnog sažimanja prosjekom na k-toj dubini F^k

$$F^k = \sum_{x,y} f_k(x, y).$$

Tada rezultate funkcije sažimanja prosjekom lineariziranom. Težine w_{ck} spajaju sve jedinice s klasom c i daju informaciju koliki je doprinos jedinice na k-toj dubini za klasu c.
Izlazna vrijednost za klasu c je tada

$$\sum_k w_{ck} \sum_{x,y} f_k(x, y) = \sum_{x,y} \sum_k w_{ck} f_k(x, y).$$

Aktivacijska mapa za klasu c se tada definira kao

$$M_c(x, y) = \sum_k w_{ck} f_k(x, y). \tag{6}$$

Vidimo kako tada $M_c(x, y)$ direktno utječe na važnost pozicije (x, y) za klasu c

$$\sum_{x,y} M_c(x, y).$$

3.3.3 Deep Learning Approach for Evaluating Knee MR Images: Achieving High Diagnostic Performance for Cartilage Lesion Detection

Sljedeći rad za koji ćemo napraviti kratki pregled je Deep Learning Approach for Evaluating Knee MR Images: Achieving High Diagnostic Performance for Cartilage Lesion Detection [5]. Kao i do sada, svrha rada je pokazati mogućnosti korištenja metoda dubokog učenja za otkrivanje oštećenog tkiva na području koljena. U radu su korištene metode segmentacije i klasifikacije. Općenito su ostvareni dobri statistički rezultati za dane podatke.
Skup podataka koji sadrži snimke s magnetske rezonance koljena je skupljen od 175 pacijenata. Za svakog pacijenta je napravljeno snimanje s tri različite tehnike u sagitalnoj ravnini. Dijagnoza za svako koljeno je dana od strane stručnih radiologa na temelju snimke sa sve tri tehnike.

Model dubokog učenja koji je ovdje primijenjen sadrži dvije konvolucijske mreže. Prva konvolucijska mreža služi za segmentaciju slike, a druga za klasifikaciju slike. Ulazni podaci na obje mreže su dvodimenzionalni.
Segmentacijska konvolucijska mreža je konstruirana koristeći enkoder-dekoder arhitekturu. Prvi dio mreže, enkoder, koristi Visual Geometry Group 16 (VGG16) konvolucijsku mrežu treniranu za prepoznavanje slika iz ImageNet Large Scale Recognition Challenge skupa podataka. Enkoder ulaznu sliku transformira u vektor značajki koji prosljeđuje dekoderu. Dekoder u ovom slučaju koristi istu VGG16 konvolucijsku mrežu. Uloga dekodera je iz vektora značajki generirati segmentiranu sliku koja odgovara ulaznoj slici. Segmentacija je napravljena tako da su intenziteti piksela podijeljeni u pet kategorija: 0-pozadina, 1-bedrena kost, 2-hrskavica oko bedrene kosti, 3- cjevanica, 4-hrskavica oko cjevanice.
Dodan je sloj sa softmax funkcijom kako bi svakom pikselu pridružili vjerojatnost pripadanja nekoj od pet navedenih klasa.
Proces učenja segmentacije je kontroliran koristeći ručno segmentirane slike koljena. Dice-ov[3] koeficijent je korišten kako bi se usporedila dobivena segmentacija s ručno napravljenom. Između dvije konvolucijske mreže je napravljen korak u kojem su locirali i ekstrahirali dijelove slike koji sadrže hrskavično tkivo. Tako su od 175 snimki koljena dobili 17395 manjih slika nad kojima je proveden proces treniranja i testiranja.
Konvolucijska mreža za klasifikaciju koristi ponovno VGG16 mrežu s dva potpuno povezana sloja na kraju kako bi kao izlazne varijable dobili vjerojatnosti za prisutnost oštećenog hrskavičnog tkiva.
Statističku analizu su proveli koristeći ROC[4] analizu kako bi ocijenili izvedbu modela. Kvaliteta izvedbe je ocjenjivana kriterijima osjetljivosti i specifičnosti za predikcije modela i za dijagnoze od strane liječnika.
Kako bi usporedili izvedbu modela s obzirom na liječničke dijagnoze korišteni su Youden-ov indeks i κ statistika.
Testni skup podataka je dva puta evaluiran modelom dubokog učenja, te dva puta evaluiran od strane nekoliko liječnika radiologa . AUC[5] vrijednosti za dvije evaluacije modela su 0.917 i 0.914. Osjetljivost modela je 84.1% za prvu evaluaciju, odnosno 80.5% za drugu evaluaciju, što je bolje od liječničkih rezultata. Specifičnost modela je lošija u odnosu na specifičnost liječničke dijagnoze i iznosi 85.2% za prvu, odnosno 87.9% za drugu evaluaciju. Detaljniji rezultati su dostupni u [5].
Zaključno, općenita točnost ovog modela je visoka. Nedostatak je što je specijaliziran samo

[3]Sørensen–Dice koeficijent za statističko mjerenje sličnosti dva uzorka
[4]Receiver operating characteristics
[5]Area under the curve

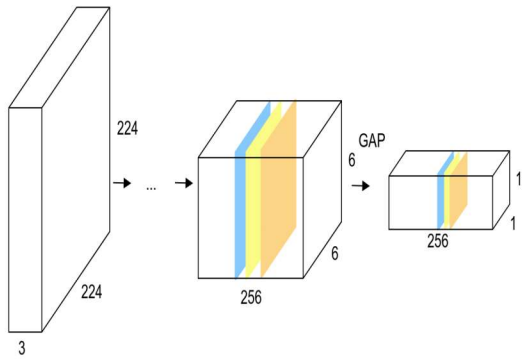
na oštećenja bedrenične kosti i cjevanice te na mali broj tehnika snimanja. Također, pristup problemu koristeći segmentaciju se pokazao dobrim, no segmentiranje slika nadziranim učenjem može biti neefikasno.

3.4 Implementacija rješenja

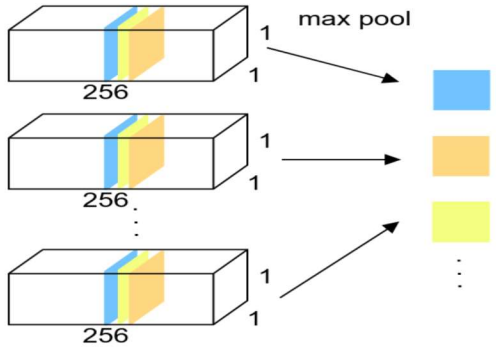
U ovom poglavlju će biti predstavljeno jedno rješenje za navedeni problem otkrivanja i klasifikacije ozljeda koljena na temelju snimki s magnetske rezonance. Problemu je, kao i do sada pristupljeno koristeći metode dubokog učenja. U nastavku će biti predstavljeni podaci dostupni za ovaj projekt te njihova priprema i obrada. Nakon toga slijedi opis konvolucijske mreže koja je korištena i procesa treniranja i validacije. Na kraju će biti pokazani i analizirani dobiveni rezultati.

3.4.1 Model

Model konvolucijske mreže jednak je onome u radu [1]. Implementacija je napravljena koristeći programski paket PyTorch. U procesu treniranja koji je izveden u 40 epoha korišten je optimizacijski algoritam Adam [6] kojemu je zadana stopa učenja s vrijednosti 10^{-5}. Osim toga, u procesu treniranja je korišten *scheduler* - modul u PyTorchu koji nudi nekoliko metoda za prilagođavanje stope učenja s obzirom na broj epoha. Ovdje je konkretno korištena metoda ReduceLROnPlateau koja radi tako da smanji stopu učenja kada se vrijednost funkcije pogreške prestane smanjivati. Tijekom procesa treniranja model je evaluiran u svakoj epohi na podacima za treniranje, a zatim na podacima za validaciju. Korištena je BinaryCrossEntropy funkcija kao funkcija pogreške. Težine modela se spremaju svaki put kada se postigne nova najmanja vrijednost funkcije pogreške na validacijskom skupu te se time sprječava prenaučenost modela.



(a) Grafički prikaz konvolucijskih slojeva u AlexNet-u na koje je dodana funkcija sažimanja prosjekom.



(b) Grafički prikaz djelovanja sažimanja maksimumom na tenzore jednog volumena kako bi se dobio jedan tenzor.

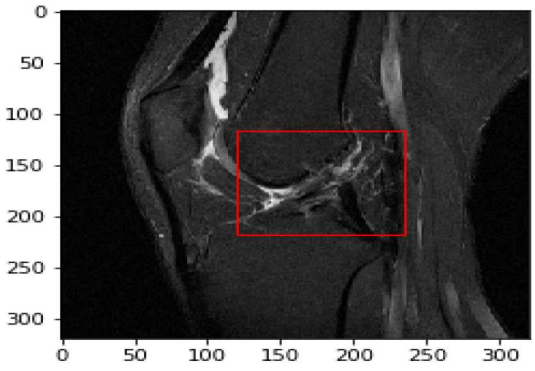
Slika 13: Grafički prikaz modela konvolucijske mreže korištenog u praktičnom projektu.

Podaci	KBC Rijeka	Opća bolnica Požega
Broj uzoraka	917	879
Broj uzoraka s ozlijedama	227	349

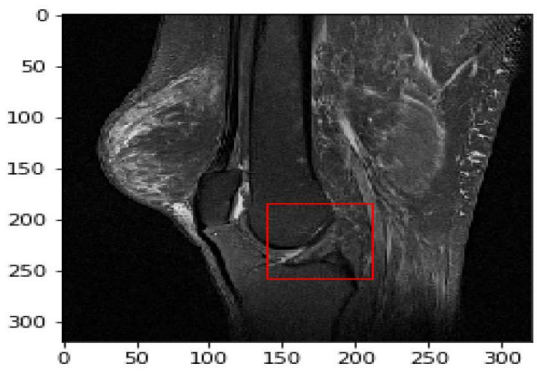
Tablica 1: Broj podataka s obzirom na podrijetlo i na dijagnozu.

3.4.2 Podaci

Snimke magnetske rezonance korištene u ovom projektu dolaze iz dva različita izvora. Prvi skup podataka je *kneeMRI dataset* iz Kliničkog bolničkog centra Rijeka[6], drugi skup podataka dolazi iz Opće županijske bolnice Požega.
Skup podataka iz Kliničkog bolničkog centra Rijeka sadrži 917 volumnih snimki koljena napravljenih na uređaju Siemens Avanto 1.5T tehnikom proton density (PD). Svaki uzorak je označen s obzirom na dijagnozu prednjeg križnog ligamenta. Oznake su: 1 - zdrav ligament, 2 - djelomično ozlijeđen ligament, 3 - u potpunosti ozlijeđen ligament. Dodatno je na svakom volumenu ručno označen pravokutnik tzv. ROI (Region Of Interes) unutar kojeg se najbolje vidi prednji križni ligament i njegovo stanje.
Skup podataka iz Opće županijske bolnice Požega sadrži 879 snimki magnetske rezonance koljena. Sve snimke su dobivene tehnikom proton density (PD), ali razlikuju se s obzirom na korištenje tehnika potiskivanja signala masti (*engl.* fat suppression).



(a) Prikazano koljeno ima oštećenje na prednjem križnom ligamentu.



(b) Uzorak zdravog koljena.

Slika 14: Uzorak snimke koljena iz skupa podataka iz Kliničkog bolničkog centra Rijeka. Crvenim pravokutnikom je obrubljeno područje interesa tzv. ROI (*engl.* Region of interest)

[6]http://www.riteh.uniri.hr/ istajduh/projects/kneeMRI/

3.4.3 Primjena aktivacijskih mapa

Na temelju predloženog postupka u radu [10], opisanog u poglavlju 3.3.2 implementirane su aktivacijske mape kako bismo potvrdili da konvolucijska mreža uči bitne značajke sa slika. Formula (6) koja glasi

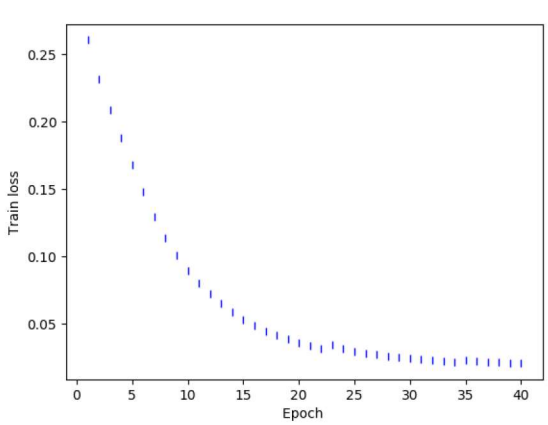
$$M_c(x, y) = \sum_k w_{ck} f_k(x, y),$$

je korištena za računanje vrijednosti aktivacijske mape. Kako bismo dobili bolju razlučivost implementirani su filteri veličine $256 \times 13 \times 13$, odnosno za razliku od navedenog rada, računanje je pomaknuto za jedan sloj unatrag. Nakon toga su izračunate aktivacijske mape stavljene preko originalnih slika. Dobiveni rezultati su potvrdili pretpostavke o bitnim značajkama snimki. Dijelovi slike na kojima su aktivacijske mape najistaknutije odgovaraju dijelovima koljena na temelju kojih se donose predikcije.

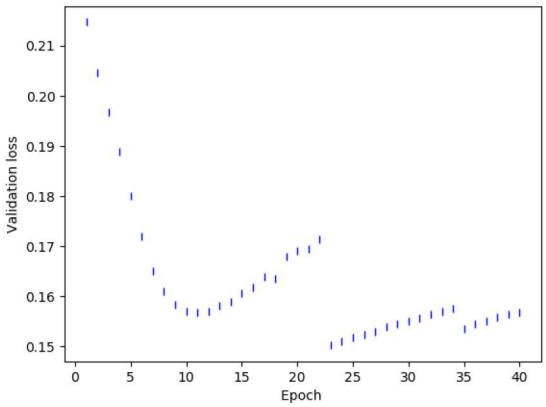
Slika 15: Primjer djelovanja aktivacijskih filtera na sliku koljena. Uočavamo kako tamnije i intenzivinije nijanse crvene odgovaraju područjima koji naviše doprinose u procesu klasifikacije.

3.4.4 Rezultati

Model je prvo treniran na skupu podataka iz Kliničkog bolničkog centra Rijeka. Proces učenja modela je izvršen u 40 iteracija. Tijekom procesa učenja je izvršena validacija funkcije pogreške kako bi se spriječila prenaučenost. Posljednje spremljene vrijednosti modela su nakon 23 iteracije. Kretanje vrijednosti funkcije pogreške na oba skupa podataka su prikazani na Slici 16. Možemo primijetiti kako vrijednosti funkcije pogreške na skupu za treniranje u skoro svim iteracijama opadaju, dok na skupu za validaciju to nije slučaj.



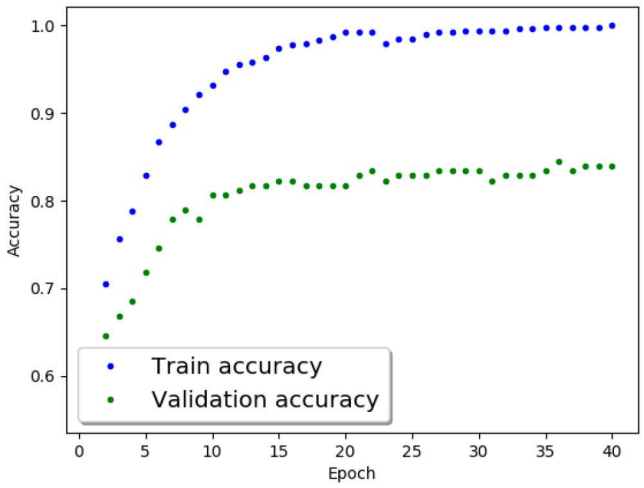
(a) Funkcija pogreške na trening skupu podataka.



(b) Funkcija pogreške na skupu podataka za validaciju.

Slika 16: Grafički prikaz vrijednosti funkcije pogreške u ovisnosti o iteracijama.

Model je tijekom učenja postigao točnost od 0.9796 na skupu podataka za treniranje i 0.8232 na skupu podataka za validaciju. Na slici 17 je graf ovisnosti točnosti modela s obzirom na epohe tijekom procesa učenje. Model je zatim testiran na novom podskupu podataka iz Kliničkog bolničkog centra Rijeka te je postignuta točnost 0.967 i vrijednost funkcije pogreške 0.033.
Nakon toga je provedeno testiranje modela na podacima iz Opće bolnice Požega gdje je model postigao točnost od 0.614 s vrijednošću funkcije pogreške 0.374. U sljedećem koraku, provedeno je treniranje modela na podacima iz Opće bolnice Požega pri čemu je dobivena točnost 0.896 na trening skupu podataka i 0.784 na validacijskom skupu.



Slika 17: Grafički prikaz točnosti modela u ovisnosti o iteracijama.

Zaključak

Koristeći konvolucijske mreže i njihovo svojstvo brzog učenja i prepoznavanje uzoraka može se razviti koristan alat za dijagnosticiranje ozljeda koljena, ali i općenito drugih ozljeda i nepravilnosti. Upotreba takvog alata u medicini može ubrzati postupke liječenja i smanjiti broj krivo procijenjenih ozljeda.

Primjenom metode učenja prenošenjem je iskorišteno znanje postojećih modela konvolucijskih mreža za prepoznavanje općenitih uzoraka, a treniranje na velikom skupu podataka koji dolaze iz različitih izvora su doprinijeli finom podešavanju parametara i prepoznavanju specifičnih uzoraka na snimkama koljena poput oblika prednjeg križnog ligamenta i meniskusa.

Smatram kako ovaj model ima potencijal za praktičnu upotrebu i kako može doprinijeti efikasnom liječenju. Isto tako, postoji prostor za daljnje nadograđivanje i modifikaciju ovog modela kako bi ga se optimiziralo i proširilo područje djelovanja.

Literatura

[1] N. Bien, P. Rajpurkar , R. L. Ball, J. Irvin, A. Park, E. Jones, M. Bereket, B. N. Patel, K. W. Yeom, K. Shpanskaya, S. Halabi, E. Zucker, G. Fanton, D. F. Amanatullah, C. F. Beaulieu, G. M. Riley, R. J. Stewart, F. G. Blankenberg, D. B. Larson, R. H. Jones, C. P. Langlotz, A. Y. Ng , M. P. Lungren *Deep-learning-assisted diagnosis for knee magnetic resonance imaging: Development and retrospective validation of MRNet.* November 27, 2018https://doi.org/10.1371/journal.pmed.1002699

[2] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006.

[3] S. C. Bushong, G. Clarke. *Magnetic Resonance Imaging: Physical and Biological Principles, 4th Edition.* Mosby, 2014.

[4] J. Duchi, E. Hazan, Y. Singer. *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.* Journal of Machine Learning Research 12 (2011) 2121-2159

[5] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning.* MIT Press, 2016.

[6] D. P. Kingma, J. L. Ba. *Adam: A method for stochastic optimization.* 3rd International Conference for Learning Representations, San Diego, 2015.

[7] A. Krizhevsky, I. Sutskever, G. E. Hinton *ImageNet Classification with Deep Convolutional Neural Networks* Advances in neural information processing systems, 2012., 1097-1105

[8] F. Liu, Z. Zhou, A. Samsonov, D. Blankenbaker, W. Larison, A. Kanarek, K. Lian, S. Kambhampati, R. Kijowski. *Deep Learning Approach for Evaluating Knee MR Images: Achieving High Diagnostic Performance for Cartilage Lesion Detection.* Radiology. 2018 Oct;289(1):160-169. doi: 10.1148/radiol.2018172986. Epub 2018 Jul 31.

[9] R. Scitovski, N. Truhar, Z. Tomljanović. *Metode optimizacija.* Sveučilište Josipa Jurja Strossmayera u Osijeku Odjel za matematiku, 2014.

[10] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba *Learning deep features for discriminative localization* Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference, 2921–2929.

[11] PyTorch Tutorials, https://pytorch.org/tutorials/

Sažetak

U ovom diplomskom radu je objašnjen pojam neuronskih i konvolucijskih mreža. Njihovo djelovanje se definira matematičkim izrazima i formulama. Ulazni podaci u neuronskim mreža prolaze kroz niz slojeva pri čemu se transformiraju i kao rezultat dobijemo vrijednosti koje dalje interpretiramo sukladno početnom problemu. Svaki model neuronskih i konvolucijskih mreža ima parametre čije vrijednosti utječu na točnost predikcija. U svrhu povećavanje točnosti, parametri se podešavaju u procesu učenja. Pristup procesu učenja je, u suštini, matematička optimizacija nekih nelinearnih funkcija koje ovise o tim parametrima. Stoga se koriste neki standardni algoritmi za optimizaciju, ali isto tako postoje algoritmi razvijeni specifično za probleme dubokog učenja. Kako bi se povećala kvaliteta modela, potrebno je provesti postupke obrade podataka prije samog učenja i regularizacije.
Motiv za kreiranjem kvalitetnih mreža za duboko učenje je njihova praktična upotreba u raznim znanstvenim granama i djelatnostima. Jedna od primjena je u medicini, za klasifikaciju slika s magnetske rezonance, gdje bi dobro naučeni model mogao pridonijeti efikasnijem dijagnosticiranju ozljeda i bolesti.

Ključne riječi: Neuronske mreže, konvolucijske mreže, strojno učenje, magnetska rezonanca, ozljede prednjeg križnog ligamenta, segmentacija slike.

Summary

This master thesis describes and presents the concept of neural and conventional networks. The basic flow of actions in those networks is defined by mathematical expressions and formulas. Input data are being transformed while flowing through the network and the obtained output is then used to form an answer to the initial problem. Network parameters are crucial for the process of learning and their values directly effect precision. In order to obtain better performance parameters are being tuned in the learning process which is basically the optimization of some nonlinear functions with respect to the mentioned parameters. Optimization is using standard algorithms from numerical mathematics, but there are also special algorithms which are results of optimization in deep learning. Building a deep learning model includes preprocessing data and regularization.
High-quality deep learning networks are useable in various scientific disciplines and businesses. One of the widespread application is for medical diagnosis and medical imaging classification. This kind of model can contribute to decision-making in a way that it gives automated diagnosis and localization of abnormalities.

Keywords: Neural networks, convolutionaln networks, machine learning, magnetic resonance, anterior cruciate ligament (ACL) tears, image segmentation.

Životopis

Danijela Jaganjac rođena je 17.12.1994. godine u Osijeku gdje završava osnovnu školu i I. gimnaziju. Nakon toga obrazovanje nastavlja na Odjelu za matematiku Sveučilišta Josipa Jurja Strossmayera u Osijeku na preddiplomskom studiju matematike kojeg završava 2017. godine sa završnim radom iz područja numeričke matematike - Gaussove kvadraturne formule za numeričku integraciju. Iste godine na Odjelu upisuje diplomski studij, smjer Matematika i računarstvo. Tijekom diplomskog studija obavlja studentski posao u tvrtki Gideon Brothers te studentsku praksu u tvrtki Mono d.o.o.