SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6344

Klasifikacija medicinskih slika konvolucijskim modelima

Ivan Almer

Zagreb, lipanj 2019.

Zahvaljujem Siniši Šegviću, svojem mentoru, na podupiranju i motivaciji tijekom izrade završnog rada, te svojoj obitelji i prijateljima na bezuvjetnoj podršci.

SADRŽAJ

1. Uvod

Jedno od vrlo važnih područja primjene računalnog vida je medicinska obrada slike. Ovo područje bavi se izvlačenjem podataka iz slika s ciljem izrade dijagnoze pacijenta. Informacije koje se mogu izvući iz takvih slika su, na primjer, otkrivanje tumora, arterioskleroze itd. Takvi problemi se možda za liječnike čine trivijalni, no naravno da se može dogoditi da niti oni mogu pogriješiti ili previdjeti neki dio, koji računalu možda ne bi promaknuo.

Klasifikacija slika je jedan od problema računalnog vida čija je zadaća na temelju ulaznog skupa podataka odrediti ispravnu klasu kojoj slika pripada. Taj je problem jedan od glavnih problema u svojem području, upravo zato što se i neki drugi problemi računalnog vida mogu svesti na klasifikaciju slika (npr. segmentacija slike).

Postoji više pristupa rješavanju problema klasifikacije slika, no pokazalo se da konvolucijske neuronske mreže rade dobar posao u segmentaciji, detekciji te prepoznavanju objekata i regija na slikama. Pojava dubokih konvolucijskih mreža je započela novo područje unutar strojnog učenja koje se zove duboko učenje. U posljednjih nekoliko godina bilježi se velika uspješnost i napredak tog područja. Jedne od najboljih mreža koje trenutno postoje su rezidualne mreže koje je osmislio Microsoftov istraživački tim sa Kaimingom Heom na čelu. Razvoj tih mreža omogućio je to da se povećavanjem dubine mreže, performanse povećavaju što nije morao biti slučaj kod ostalih dubokih konvolucijskih mreža. U sklopu ovog rada, koristio sam upravo te mreže kako bih izvršio klasifikaciju slika.

Cilj ovog rada je što ispravnije klasificirati slike kožnih oboljenja iz skupa podataka HAM:10000. U nastavku slijedi općenito o umjetnim neuronskim mrežama i konvolucijskim mrežama, a zatim ću opisati izbor metoda te dijelove programske implementacije. U zadnjem dijelu ću prikazati rezultate i navesti mjesta na kojima bi se mogla napraviti poboljšanja.

2. Umjetne neuronske mreže

2.1. Motivacija razvoja umjetnih neuronskih mreža

Ljudski mozak je najsloženiji organ u ljudskom tijelu koji je zapravo odgovoran za održavanje svih funkcija u našem tijelu. Osim toga mozak ima sposobnost pamćenja, učenja, donošenja odluka itd. Nezamislivo je koliki je protok informacija u našem mozgu tijekom samo jedne sekunde.

Poznato je da se ljudski mozak sastoji od izuzetno velikog broja neurona koji rade paralelno. Prilikom prijenosa informacije između neurona, svaki neuron dobije informacije od prosječno 1000 do 10000 drugih neurona. Tako dok gledamo u neku sliku, mozak interpretira informacije dobivene kroz mrežu neurona i može, primjerice prepoznati da se na toj slici radi o određenoj životinji. Takvo ponašanje mozga nam nije interpretabilno, odnosno prezahtjevno je za razumjeti, te ne znamo kako bismo napisali algoritam koji bi mogao, primjerice iz slike prepoznati radi li se o slici psa ili je to možda slon. To je navelo ljude da pokušaju razviti računalni sustav koji će djelovati na jednak način kao ljudski mozak. Ideja je izgraditi sustav koji bi mogao učiti na primjerima. Umjetne neuronske mreže (engl. *artificial neural networks*) mogu se koristiti za probleme klasifikacije te za probleme regresije.

2.2. Neuron

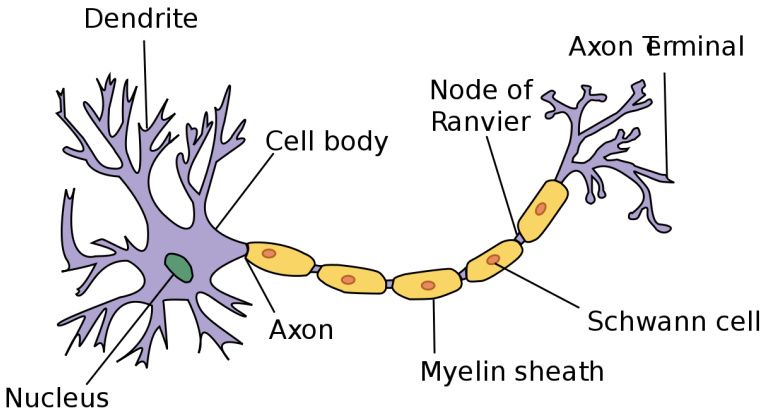
Biološki neuron je živčana stanica koja je temeljni građevni element živčanog tkiva. Neuron je električki podraživa stanica koja promjenama u naponu preko sinapsi komunicira s drugim neuronima i prenosi informacije. Sastavni dijelovi neurona:

1. soma (tijelo stanice) - sadrži jezgru i to je mjesto gdje se odvija većina sinteze proteina,

2. dendriti - kraći produžeci koji s osjetnih organa ili drugih živčanih stanica dovode živčano uzbuđenje na tijelo stanice,

3. akson - duži produžetak neurona kojemu je funkcija prenositi impulse s tijela stanice na druge živčane stanice ili izvršne organe,

4. završni članci - članci koji sadrže sinapse kojima se impulsi prenose na druge neurone



Slika 2.1: Biološki neuron.

Svaki neuron impulse prima dendritima i prenosi ih dalje kroz tijelo, akson i konačno na završne članke koji prenose te impulse dalje na sljedeće neurone.

Inspiracijom modela biološkog neurona može se modelirati umjetni neuron koji na svojem ulazu dobiva niz vrijednosti i proizvodi izlaz. Matematički gledano, umjetni neuron je funkcija koja svaki od ulaza $x_1, x_2, ...x_n$ množi s njegovom težinom w_i (engl. *weight*) i dodaje se slobodni član w_0 (engl. *bias*). Konceptualno svaki od ulaza u neuron odgovara jednom dendritu. Tijelo neurona agregira vrijednosti od svih ulaza, sumira ih i tome dodaje slobodni član. Ta suma čini srž onoga što će biti izlaz iz neurona:

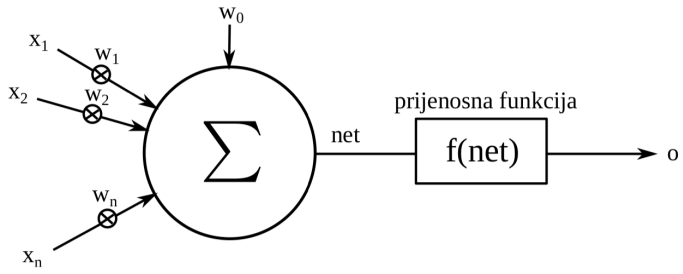
$$net = \sum_{i=1}^{n} x_i w_i + w_0 \tag{2.1}$$

Budući da vrijedi da je $w_0 = w_0 * 1$, dogovorno se na ulaz x_0 dovodi 1 s ciljem da se gornja formula zapiše u obliku:

$$net = \sum_{i=0}^{n} x_i w_i \tag{2.2}$$

Svaki neuron ima svoju aktivacijsku funkciju te je konačan izlaz iz neurona vrijednost koja se dobije primjenjivanjem aktivacijske funkcije na izračunatu sumu.

$$output = f(net) = f(\sum_{i=0}^{n} x_i w_i) \tag{2.3}$$

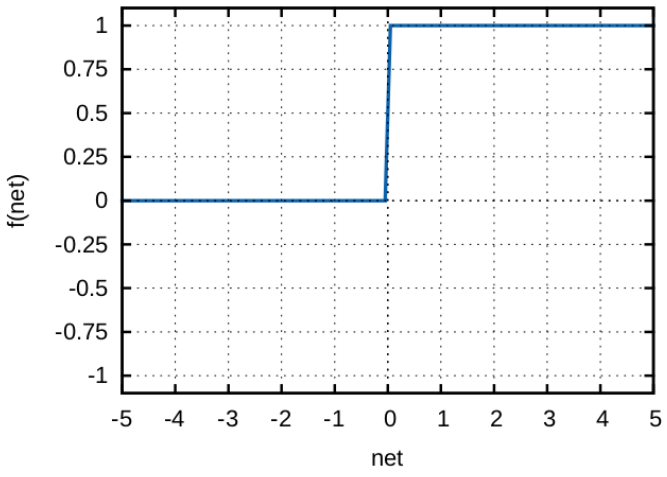


Slika 2.2: Umjetni neuron.

2.3. Aktivacijske funkcije

Aktivacijska funkcija umjetnog neurona definira izlaz iz neurona za zadane ulazne podatke. Obično predstavlja abstrakciju ispaljivanja biološkog neurona. U svojoj najjednostavnijoj formi funkcija izgleda binarno, točnije to je step funkcija.

$$f(net) = \begin{cases} 0 & n < 0 \\ 1 & else \end{cases} \tag{2.4}$$



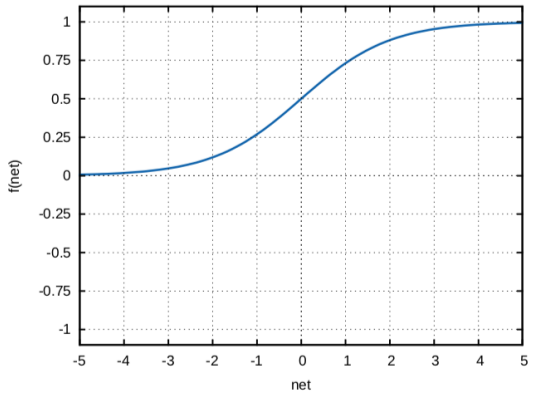
Slika 2.3: Step funkcija.

Aktivacijska funkcija koja je povijesno najvažnije je sigmoidalna funkcija. Modeli koji su najviše koristili sigmoidalnu aktivacijsku funkciju su oni modeli koji predviđaju neke vjerojatnosti, jer je vrijednost sigmoide uvije u intervalu $(0, 1)$. Sigmoidalna funkcija je i derivabilna iako vrijednosti derivacije brzo konvergiraju k 0 kada $x \to \infty$ i kada $x \to -\infty$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Derivacija sigmoidalne funkcije jest

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x)) \tag{2.6}$$



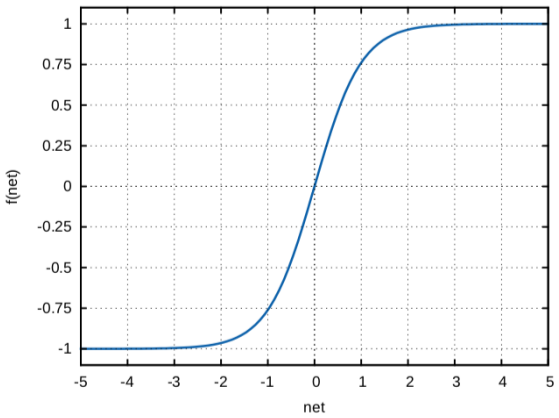
Slika 2.4: Sigmoidalna funkcija.

Još jedna, također derivabilna, aktivacijska funkcija je tangens hiperbolni. On je baš kao i sigmoidalna funkcija ograničen, no granice funkcije su $[-1, 1]$. Prednost takvog pristupa je ta da će negativne vrijednosti biti mapirane strogo negativno, a vrijednosti koje se nalaze oko 0 će biti mapirane blizu 0 na grafu $tanh$.

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.7}$$

Njegova je derivacija jest:

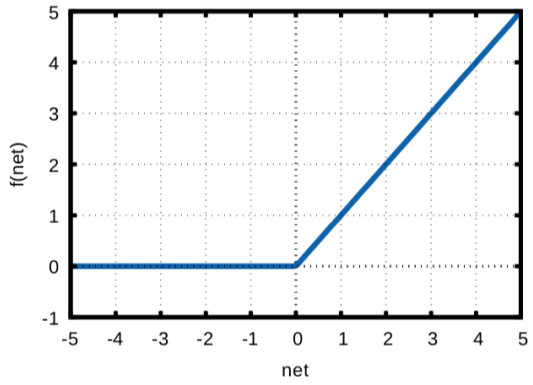
$$\frac{d}{dx}tanh(x) = 1 - tanh^2(x) \tag{2.8}$$



Slika 2.5: Tangens hiperbolni.

Aktivacijska funkcija[Vinod Nair (2010)] koja je ranih 2010-ih mnogo doprinijela razvoju dubokih modela jest zglobnica (engl. *extitRectified Linear Unit, ReLU*). I funkcija i njena derivacija su monotone. ReLU je definirana kao:

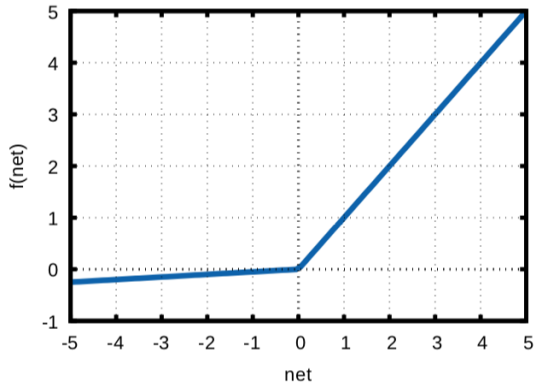
$$ReLU(x) = max(x, 0) \tag{2.9}$$



Slika 2.6: ReLU.

Međutim problem sa ReLU funkcijom nastaje zato što se sve vrijednosti manje od 0 direktno mapiraju u 0. Iz tog razloga postoji mala dorada funkcije ReLU koja se naziva propusna zglobnica[Xu et al. (2015)] (engl. *Leaky ReLU*) jer s nekim malim parametrom α propušta negativne vrijednosti da budu negativne, a ne da budu mapirane u 0. Na taj se način rješava problem umirućeg ReLU-a (engl. *A dying ReLU problem*) čiji je gradijent za vrijednosti manje od 0 uvijek jednak 0.

$$LeakyReLU(x) = \begin{cases} x & x >= 0 \\ \alpha x & else \end{cases} \tag{2.10}$$



Slika 2.7: Leaky ReLU.

2.4. Strojno učenje

Strojno učenje je područje umjetne inteligencije čiji je cilj omogućiti računalima da samostalno uče. Omogućava nam otkrivanje i prepoznavanje uzoraka iz podataka, kao i predikciju na temelju ulaznih podataka. Ono što je bitno spomenuti za strojno učenje jest to da programi nemaju neka preddefinirana pravila, već sama uče pronaći ta pravila u podacima.

Postoje 2 vrste strojnog učenja: nadzirano i nenadzirano učenje (engl. *supervised and unsupervised learning*)

U nadzirano učenje spadaju problemi učenja kod kojih znamo točne očekivane izlazne vrijednosti. Točnije nadzirano učenje rješava probleme npr., pronalaženja uzoraka u podacima, umjesto nas. Cilj je što točnije predvidjeti izlaznu vrijednost na podacima za koje znamo točan izlaz, kako bi takav sustav mogao dobro predviđati izlazne vrijednosti i za podatke koje nikada nije vidio.

U nenadzirano učenje je spadaju problemi kod kojih za ulazne podatke nemamo određene izlazne vrijednosti. Cilj nenadziranog učenja je pronaći određene pravilnosti u podacima. Nenadzirano učenje može se podijeliti u sljedeće vrste problema:

- grupiranje (engl. *clustering*) - pronalaženje grupacija u podacima
- otkrivanje stršećih vrijednosti (engl. *outlier detection*) - otkrivanje vrijednosti koje odudaraju od ostatka bez da znamo kako one zapravo izgledaju
- smanjenje dimenzionalnosti (engl. *dimensionality reduction*) - uklanjanje značajki ulaznih podataka koje su redundantne

Jezgra računalnog programa koji se temelji na strojnom učenju jest model. Model strojnog učenja može biti interpretiran kao crna kutija koja za određeni ulaz daje izlaz. U strojnom učenju postoje razni modeli, poput stable odluke, modela linearne

regresije, neuronskih mreža itd.

Modeli se mogu trenirati i evaluirati te su oni odgovorni za ono što ljudi poimaju inteligentnim odlukama. Njihov odabir svakako ovisi o problemu koji se rješava.

2.4.1. Gubitak

Modeli strojnog učenja uče tako da se njihovo trenutno znanje na određeni način evaluira i na temelju toga podešava ne bi li greška bila što manja. Kao mjera kazne za loše izlazne vrijednosti iz modela koristi se funkcija gubitka. Funkcija gubitka je to veća što je greška izlaza modela za određeni podatak veća - dakle cilj je da model ima što manji gubitak.

Naravno, funkcija gubitka uvelike ovisi o problemu kojeg model rješava, no postoje neke značajke na temelju kojih odabiremo funkciju gubitka, kao na primjer, lakoća izračunavanja derivacije funkcije gubitka.

Široko gledano funkcije gubitka mogu se podijeliti u funkcije gubitka regresije, te funkcije gubitka klasifikacije. Kod klasifikacije želimo da model može predvidjeti izlaz na temelju ulaznog podatka, gdje je izlaz kategorička vrijednost iz nekog konačnog skupa vrijednosti. Regresija se, s druge strane, bavi predviđanjem neke kontinuirane vrijednosti na temelju ulaznih podataka.

Neke funkcije gubitka prikazane su u nastavku.

Srednja kvadratna pogreška (engl. *Mean Square Error*)

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{2.11}$$

Kao što samo ima govori, ova funkcija gubitka zbraja sva kvadratna odstupanja predviđenih vrijednosti od stvarnih vrijednosti. Posljedica toga je da ona predviđanja koja su daleko od stvarne vrijednosti bivaju jako kažnjena i uvelike mogu doprinijeti vrijednosti greške.

Unakrsna entropija (engl. *Cross-Entropy Loss*)

Funkcija gubitka unakrsna entropija za jedan podatak glasi:

$$L = -\mathbf{y} log(\hat{\mathbf{y}}) \tag{2.12}$$

gdje je $\mathbf{y}$ jedinično kodirani vektor stvarnih razreda ulaznog podatka, a $\hat{\mathbf{y}}$ je vektor vjerojatnosti za pojedini razred, koje je model predvidio. Budući da su u vektoru $\mathbf{y}$

sve vrijednosti 0 osim one vrijednosti koja se odnosi na točan razred, gore navedeni gubitak se svodi na:

$$L = -log(\hat{y}) \tag{2.13}$$

U ovom je slučaju $\hat{y}$ vjerojatnost da je podatak točno klasificiran i to je skalarna vrijednost. Matematički zapisano:

$$\hat{y} = logP(Y = y_i|x_i) \tag{2.14}$$

Pri čemu je x_i i-ti ulazni podatak iz minigrupe, a y_i je točna oznaka za i-ti podatak.

Funkcija gubitka unakrsna entropija sumira gore navedene gubitke za sve podatke iz minigrupe. Nakon što su sve te vrijednosti zbrojene, ukupni se gubitak cijele minigrupe podatka podijeli s veličinom minigrupe (u mojem slučaju 16).

$$loss = -\frac{1}{N}\sum_{i=0}^{N} logP(Y = y_i|x_i) \tag{2.15}$$

2.4.2. Optimizacija

Optimizacija je postupak optimizacije funkcije cilja (npr. minimizacije funkcije gubitka) kako bi se poboljšao model koji u sebi ima određene parametre koje je moguće naučiti i koji su zapravo odgovorni za izlaz modela kada mu je dan neki ulazni podatak. Minimizacija funkcije gubitka odvija se treniranjem modela, što zapravo nije ništa drugo nego podešavanje parametara modela s ciljem dobivanja što točnijeg izlaza iz modela.

Postoje razne metode optimizacije, poput Adam optimizatora ili gradijentnog spusta i njegovih inačica, no u ovom ću se poglavlju fokusirati samo na gradijentni spust.

Gradijentni spust (engl. *Gradient Descent*)

Gradijentni spust je metoda optimizacije koja se temelji na kretanju u smjeru negativnog gradijenta s ciljem postizanja optimuma. Vrijednost bilo koje funkcije oko neke određene točke može se prikazati Taylorovim redom.

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots \tag{2.16}$$

Gore prikazana formula točno reprezentira funkciju oko točke a, no u praksi nam nije potrebno da ta reprezentacija bude potpuno točna nego ćemo se zadovoljiti s dovoljno dobrom aproksimacijom. Bitno je spomenuti da su doprinosi članova Taylorovog reda s velikim n prilično mali i greška aproksimacije shodno tome nije velika. Kod gradijentnog spusta uključeni su prvi i drugi član, točnije posljednji član koji je uključen je član koji sadrži prvu derivaciju funkcije. U nastavku je prikazan Taylorov red oko točke x za pomak Δx.

$$f(x + \Delta x) = f(x) + \frac{df(x)}{dx}\Delta x = f(x) + \nabla f(x)^{\mathrm{T}}\Delta x \tag{2.17}$$

Kako je već ranije spomenuto ova se metoda optimizacije temelji ne kretanju u smjeru negativnog gradijenta se, shodno tome, u pomak uvrštava pomak u tom smjeru, gdje je g gradijent funkcije gubitka:

$$\Delta x = -\varepsilon \cdot g \tag{2.18}$$

$$\boldsymbol{g} = \nabla f(\boldsymbol{x}) \tag{2.19}$$

Nakon uvrštavanja prethodnih formula u Taylorov red s 2 člana, dobijemo sljedeći izraz:

$$f(x - \varepsilon \cdot \boldsymbol{g}) = f(\boldsymbol{x}) - \varepsilon \cdot \boldsymbol{g}^{\mathrm{T}}\boldsymbol{g} \tag{2.20}$$

Gradijentni spust ima hiperparametar koji je veoma bitan za postupak optimizacije, a to je stopa učenja ε. Stopa učenja kontrolira da pomak u smjeru negativnog gradijenta ne bude niti prevelik niti premalen.

Odaberemo li premalu stopu učenja funkcija će se sporo primicati optimumu, jer što se bliže primičemo to je gradijent funkcije bliži nuli, te još pomnožen s malom stopom učenja daje neznatan pomak. U slučaju da odaberemo preveliku stopu učenja postupak bi mogao početi divergirati, te ne bismo uspjeli optimirati funkciju gubitka.

Iz tih je razloga bitan dobar odabir stope učenja i u slučaju da jest dobar postupak će se svakom iteracijom algoritma primicati malo po malo optimumu, no naravno to ne isključuje mogućnost da postupak zapne u nekom lokalnom optimumu. Tom problemu može se doskočiti, primjerice odabirom funkcije gubitka koja je konveksna na cijeloj svojoj domeni.

Osim stope učenja postoji i određeni uvjet zaustavljanja u algoritmu, a to može biti maksimalan broj iteracija prije nego se prekine optimizacija, no također i određena blizina optimalnoj točki (engl. *threshold*).

U nastavku se nalazi pseudokod algoritma gradijentnog spusta.

Algorithm 1 Gradijentni spust

Ulaz: Stopa učenja ε

Ulaz: Početna točka x

$k := 1$

while nije ispunjen uvjet zaustavljanja **do**

izračunaj gradijent: $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$

napravi ažuriranje: $\boldsymbol{x} \leftarrow \boldsymbol{x} - \epsilon \boldsymbol{g}$

$k := k + 1$

end while

2.5. Klasifikacija

U strojnom učenju i statistici, klasifikacija je postupak pridjeljivanja oznaka uzorcima na temelju značajki uzorka (npr. oblik, boja itd.). Cilj je izgraditi klasifikatorski sustav koji na temelju uzoraka iz skupa za učenje ima mogućnost ispravno klasificirati nove uzorke koje još nije "vidio".

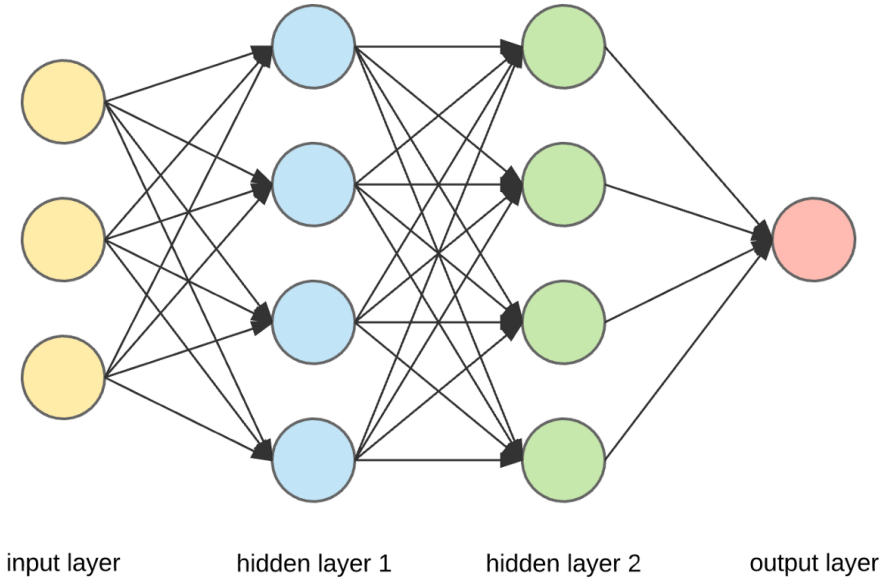
U slučaju kada postoje samo 2 razreda radi se o binarnoj klasifikaciji. U takvim slučajevima izlaz ima samo 2 jasno razlučiva stanja (npr. 0 i 1). Ako imamo više razreda obično se koristi jednojedinično kodiranje (engl. *one hot encoding*), što znači da izlaza ima onoliko koliko ima razreda i i-ti izlaz je jednak 1 u slučaju kada ulazni podatak pripada i-tom razredu, a inače je 0.

2.6. Arhitektura neuronske mreže

Pojam arhitekture neuronske mreže govori kako su neuroni međusobno povezani i koliko ih ima. Kod neuronskih mreža postoje ulazni sloj, određeni broj skrivenih slojeva te izlazni sloj. Ulazni i izlazni sloj su jedina poveznica mreže s vanjskim svijetom, jer su skriveni slojevi povezani samo s idućim slojem bez neke druge s vanjskim elementima (slika u nastavku).

Potpuno povezane unaprijedne neuronske mreže imaju sljedeće karakteristike: [arc]

1. Neuroni su posloženi u slojeve, sadrže ulazni i izlazni sloj te određeni broj skrivenih slojeva



Slika 2.8: Arhitektura umjetne neuronske mreže.

2. Svaki neuron je povezan sa svakim neuronom u idućem sloju (informacija samo ide unaprijed) i to objašnjava zašto se ove mreže zovu unaprijedne

3. Ne postoje veze između neurona u istom sloju

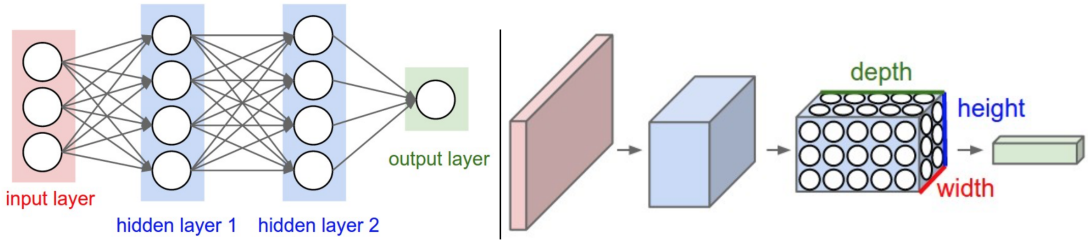
Dodavanjem sve više slojeva u mrežu i više neurona u slojeve mreža dobiva na ekspresivnosti i može naučiti složenije odnose, ali je sklona pretreniranju (engl. *overfitting*). Cilj je izgraditi neuronsku mrežu koja ima dovoljno slojeva da može prepoznati potrebne odnose koji postoje u podacima, ali želimo da to bude minimalno slojeva koliko je potrebno kako bi mreža što manje bila osjetljiva na prenaučenost.

2.7. Konvolucijske neuronske mreže

Konvolucijske mreže su umjetne neuronske mreže koje su veoma efikasne u rješavanju problema identificiranja lica, objekata, prometnih znakova itd. [con (2016)]. Bitno je spomenuti da konvolucijske mreže imaju eksplicitnu pretpostavku da ulazni podatak ima određenu topološku organizaciju (primjerice slika, matrica, tenzor ili graf).

Klasične neuronske mreže ne skaliraju dobro sa slikama, jer ako je npr. ulazni podatak slika 200x200x3 (200 piksela široka, 200 piksela visoka, 3 kanala boje), svaki neuron u prvom skrivenom sloju bi imao $200*200*3 = 120,000$ težina. Štoviše, takvih neurona neće biti malo i tako velik broj parametara bi vrlo brzo doveo do pretreniranja mreže. Dvodimenzionalni konvolucijski slojevi imaju neurone raspodijeljene u 3 di-

menzije, za razliku od klasičnih neuronskih mreža koja imaju raspodijeljene neurone u 2 dimenzije. Na sljedećoj slici prikazana je usporedba klasične neuronske mreže i konvolucijske neuronske mreže s dvodimenzionalnim konvolucijama.



Slika 2.9: Usporedba klasične i konvolucijske neuronske mreže.

Izvor: http://cs231n.github.io/convolutional-networks/

Za izradu konvolucijskih neuronskih mreža koriste se 3 glavna tipa slojeva:

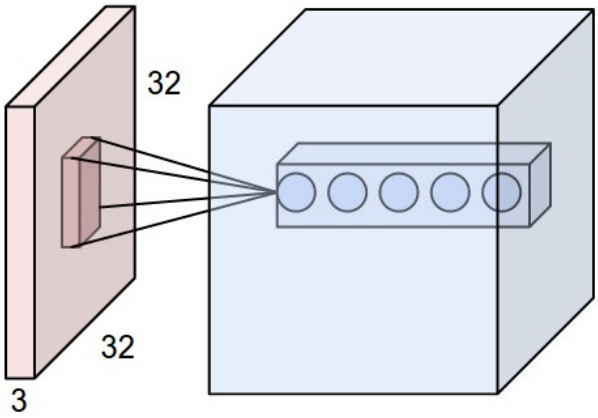
- Konvolucijski sloj (engl. *Convolutional Layer*)
- Sloj sažimanja (engl. *Pooling Layer*)
- Potpuno povezani sloj (engl. *Fully-Connected Layer*) kao kod klasičnih neuronskih mreža

2.7.1. Konvolucijski sloj (engl. *Convolutional Layer*)

To je jezgreni gradivni element konvolucijskih neuronskih mreža koji obavlja većinu računalno zahtjevnog posla. Parametri tog sloja se sastoje od skupa filtera koje je moguće naučiti. Svaki je takav filter prostorno mali (npr. 5x5x3 gdje su visina i širina 5 piksela, a 3 je dubina ulaznog tenzora odnosno broj kanala boja). Takav se filter povlači preko ulaza, računaju se skalarni produkti na svakoj poziciji filtra. Svaki filter u konvolucijskom sloju proizvede 2-dimenzionalnu aktivacijsku mapu, zatim se te aktivacijske mape od svih filtera u sloju slože (npr. 12 filtera proizvede 12 aktivacijskih mapa) i tako nastaje izlazni volumen (jer ima 3 dimenzije).

Postoje 3 hiperparametra koji kontroliraju dimenziju izlaznog volumena iz sloja. To su sljedeći:

- Dubina (engl. *depth*) - broj filtara koji se nalaze u sloju
- Korak (engl. *stride*) - korak za koji se filteri pomiču po slici, točnije broj piksela za koji skaču prilikom prolaska preko slike



Slika 2.10: Konvolucijski sloj.

Izvor: http://cs231n.github.io/convolutional-networks/

– Dopuna nulama (engl. *zero-padding*) - nekada je potrebno dodati nule na rubove slike i to nam daje mogućnost kontrolirati izlaznu dimenziju konvolucijskog sloja (često se koristi za očuvanje prostornih dimenzija)

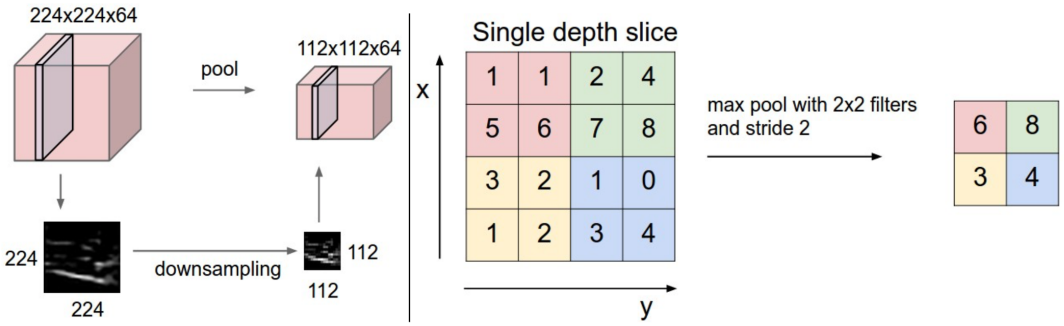
2.7.2. Sloj sažimanja (engl. *Pooling Layer*)

Sloj sažimanja se uobičajeno postavlja između konvolucijskih slojeva kako bi se smanjila prostorna veličina podatka, a s time i broja parametara koje treba naučiti, jer su prostorne dimenzije podatka manje. Iz istog se razloga smanjuje i broj računskih operacija u mreži. Na taj način slojevi sažimanja smanjuju memorijski otisak modela prilikom treniranja. Sloj sažimanja djeluje neovisno na svakoj od dubina ulaznog podatka i sažima ju, koristeći MAX ili AVG (prosjek, engl. average) operaciju. Uobičajena postavka ovog sloja je s filterima 2x2 i korakom 2. Sloj s takvim hiperparametrima smanjuje broj piksela 2 puta po širini i visini.

Na sljedećoj slici prikazan je MAX sloj sažimanja koji ima uobičajene postavke. Povlačenjem tog filtra preko ulaznog podatka na svakoj se poziciji odredi maksimalna vrijednost i to je izlazna vrijednost za taj predio ulaznog podatka. Primjerice kada se filtar nalazi na poziciji označenoj zeleno, maksimalna vrijednost u ulaznom podatku na toj poziciji jest 8 i ona predstavlja sažetak tog dijela podatka.

2.7.3. Potpuno povezani sloj (engl. *Fully-Connected Layer*)

Neuroni u potpuno povezanim slojevima imaju poveznice prema svima aktivacijama iz prethodnog sloja kao kod klasičnih neuronskih mreža. Aktivacije tih neurona se dakle



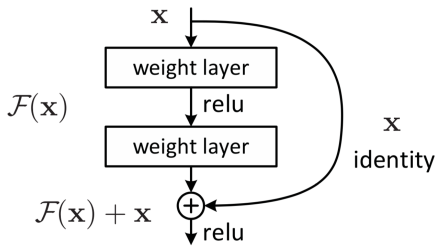
Slika 2.11: Sloj sažimanja.

Izvor: http://cs231n.github.io/convolutional-networks/

mogu izračunati množenjem matrica i dodavanjem pomaka (engl. *bias*)

2.8. Rezidualne neuronske mreže (engl. *Residual neural networks, ResNet*)

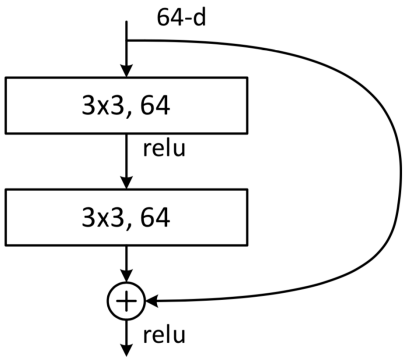
Rezidualne mreže su također konvolucijske neuronske mreže, no sadrže određena poboljšanja. Naime, rezidualne mreže sadrže određene skokove između slojeva, točnije prečace preko jednog ili više idućih slojeva. Motivacija iza toga je ta da se ponovnim korištenjem aktivacija od prijašnjih slojeva riješi problem degradacije točnosti dubljih modela prilikom dodavanja više slojeva.[Kaiming He (2015)] Danas se vjeruje da se rezidualni modeli ponašaju kao kombinacija relativno plićih modela.[Veit et al. (2016)]



Slika 2.12: Rezidualna gradivna jedinica.

Izvor: Kaiming He (2015)

Eksperimentima je pokazano da pogreška pri treniranju rezidualnih mreža pada što je mreža dublja, za razliku od klasičnih neuronskih mreža kod kojih je situacija obrnuta. Postoji nekoliko inačica rezidualnih neuronskih mreža (ResNet-18/34/50/101/152). Rezidualne mreže ResNet-18 i ResNet-34 su građene od klasičnih rezidualnih blokova koji su prikazani na sljedećoj slici:



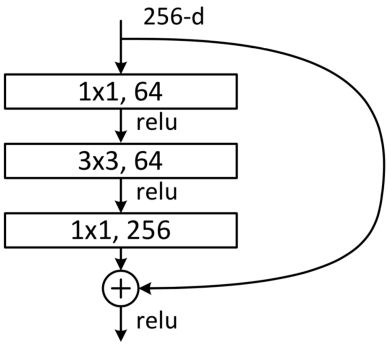
Slika 2.13: Klasična rezidualna jedinica.

Izvor: He et al. (2016)

Klasičan rezidualni gradivni blok sastoji se od 2 konvolucijska sloja 3x3 sa ReLU aktivacijskim funkcijama, dok se blokovi s uskim grlom koriste za dublje rezidualne mreže (ResNet-50, ResNet-101, ResNet-150).

Ti se blokovi sastoje od dva konvolucijska sloja 1x1 između kojih se nalazi jedan konvolucijski sloj 3x3. Odgovorni su za smanjivanje i zatim povećavanje(vraćanje) dimenzija podatka. Osim toga sadrže i, već spomenuti, skok, koji dovodi podatak s ulaza u blok na izlaz iz bloka gdje se zbraja s izlazom iz kombinacije gore spomenutih konvolucija. Takvi blokovi uspješno sprječavaju prenaučenost.[Kaiming He (2015)]

ResNet je razvio Microsoftov istraživački tim i mreža je doživjela veliki uspjeh na mnogim natjecanjima. U svojoj implementaciji klasifikacije medicinskih slika koristio sam rezidualnu mrežu ResNet-50.



Slika 2.14: Blok s uskim grlom (engl. *bottleneck*).

Izvor: Kaiming He (2015)

3. Opis podataka

Podatke za svoj rad preuzeo sam sa stranice Kaggle[1]. Skup podataka HAM10000 (Human Against Machine with 10000 training images) sastoji se od 10015 dermatoskopskih slika pigmentiranih kožnih nepravilnosti. U podacima se nalaze reprezentativni uzorci 7 različitih bolesti:

- Bowenova bolest - vrlo rana forma raka kože koja je lako izlječiva. Glavni znak je crveni ljuskasti pokrov na koži
- Karcinom bazalnih stanica - vrsta raka kože koji započinje u bazalnim stanicama (kožne stanice koje proizvode nove stanice kada stare umru)
- Benigne nepravilnosti slične keratozi - seboreične keratoze, sunčeve pjege itd.
- Dermatofibrom - benigni tumor kože koji je po svojem izgledu sličan madežu. Uzrok nastanka dermatofibroma niti dan danas nije sa sigurnošću poznat, ali može nastati kao reakcija tijela na manju traumu, ubod insekta ili uraslu dlaku.
- Melanom - zloćudni tumor pigmentiranih stanica kože. To je bolest koja najčešće može dovesti do smrti. (češće se javlja kod žena nego kod muškaraca)
- Melanocitski nevus - običan madež
- Vaskularna oštećenja - poput angioma, angiokeratoma, piogenih granuloma i krvarenja

Više od 50% kožnih nepravilnosti koje se nalaze u podacima su ustanovljene histopatološkom analizom, dok je za ostale nepravilnosti zaključak o kojoj se bolesti radi donesen na temelju naknadne analize, koncenzusa eksperata ili potvrdom mikroskopiranjem.

Metapodaci o svakoj slici, odnosno zapisu, nalaze se u jednoj csv datoteci (engl. *comma-separated values*) gdje se nalaze ne samo podaci o imenima slika i njihovima oznakama, već i o nekim dodatnim podacima o pacijentu, kao što su godine, spol te

[1] https://www.kaggle.com

mjesto na tijelu na kojem je uočena spomenuta nepravilnost. U okviru ovog rada nisam koristio te dodatne informacije, već samo sliku, te njenu oznaku.

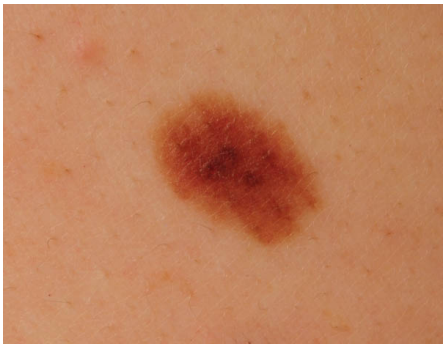
Slika 3.1: Bowenova bolest.

Slika 3.2: Karcinom bazalnih stanica.

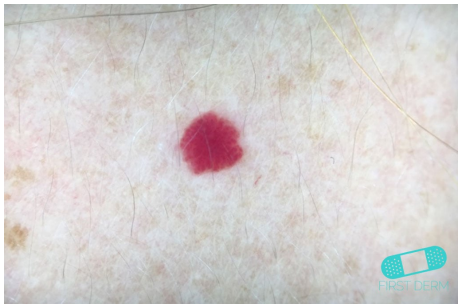
Slika 3.3: Seboroična keratoza.

Slika 3.4: Dermatofibrom.

Slika 3.5: Melanom.



Slika 3.6: Melanocitski nevus (madež).



Slika 3.7: Angiom.

4. Implementacija i rezultati

4.1. Korištene tehnologije i alati

Za izradu modela koristio sam programski jezik Python[1] u kombinaciji s bibliotekama za učitavanje i transformaciju podataka Numpy[2] te PIL.Image[3] bibliotekom za manipulaciju slikama, odnosno učitavanje i smanjivanje. Model je ostvaren u PyTorch radnom okviru. Osim toga za učitavanje metapodataka o slikama i pacijentima koristio sam Pandas[4] biblioteku koja taj postupak olakšava.

Od PyTorchevih modula koristio sam modul nn kako bih napravio izmjenu u već postojećoj ResNet-50 mreži, te modul $optim$ koji nudi implementacije mnogih optimizacijskih algoritama od kojih sam se odlučio za Adam optimizator.

4.1.1. Općenito o PyTorch biblioteci

PyTorch[5] je biblioteka za strojno učenje koja ima implementacije algoritama i gradivnih elemenata potrebnih za izradu modela strojnog učenja. Istovremeno pojednostavljuje njihovu uporabu. PyTorch je razvio Facebook-ov istraživački tim i besplatan je za korištenje te je njegova implementacija otvoreni kod (engl. *open-source*).

PyTorch efikasno provodi računanje s tenzorima (kao Numpy), no još dodatno nudi izuzetno ubrzanje izvođenjem na grafičkim karticama. Tenzori su strukture koje mogu biti tretirane kao višedimenzionalna polja te su po svojstvima slična Numpy-evim poljima (engl. *arrays*). Za razliku od Numpy-evih polja tenzori mogu biti korišteni na grafičkim karticama što bitno ubrzava izvođenje programa. Ova biblioteka sadrži 3 bitna modula, a to su $autograd$, $optim$ te nn. Modul $autograd$ odgovoran je za automatsku diferencijaciju za sve operacije nad tenzorima, točnije on sadrži potrebne

[1] https://www.python.org
[2] https://www.numpy.org
[3] https://pillow.readthedocs.io/en/3.1.x/reference/Image.html
[4] https://pandas.pydata.org
[5] https://pytorch.org

implementacije funkcija za algoritam propagacije pogreške unatrag (engl. *Backpropagation algorithm*). Modul $optim$ sadrži implementacije mnoštva različitih optimizacijskih algoritama korištenih za izgradnju neuronskih mreža. Modul nn nalazi se na malo višoj razini nego $autograd$ modul i olakšava definiranje računskih grafova i uzimanje gradijenata, jer ponekad je za kompleksne neuronske mreže $autograd$ modul na preniskoj razini te u tim situacijama nn modul može mnogo pripomoći.

Primjer definiranja slojeva jednostavne neuronske mreže u PyTorchu:

```
import torch
from torch import nn

# konvolucijski sloj koji ima 1 ulazni kanal, 6 izlaznih, filteri
conv1 = nn.Conv2d(1, 6, 3)
conv2 = nn.Conv2d(6, 16, 3)

fc1 = nn.Linear(16 * 6 * 6, 120)
fc2 = nn.Linear(120, 10)
```

Prikazani isječak koda prikazuje stvaranje 2 konvolucijska sloja i 2 potpuno povezana sloja. Prvi konvolucijski sloj $conv1$ je sloj koji ima 1 ulazni kanal, 6 izlaznih kanala, te su mu filteri 3x3. Sljedeći konvolucijski sloj $conv2$ je sloj koji ima 6 ulaznih kanala, 16 izlaznih kanala te filtere 3x3. Prvi potpuno povezani sloj $fc1$ je potpuno povezani sloj koji ima 576 ulaznih značajki (točnije očekuje tenzore koji imaju 16 kanala gdje je svaki od kanala dimenzija 6x6) te 120 izlaznih. Posljednji, $fc2$ potpuno povezani sloj, je sloj koji ima 120 ulaznih značajki te 10 izlaznih.

4.2. Predobrada i učitavanje podataka

Za učitavanje metapodataka iz ranije spomenute csv datoteke koristio sam biblioteku Pandas. Nakon učitavanja podaci su bili nasumično izmiješani. Iz svakog retka su izvučeni svi podaci, no ono što sam jedino koristio iz tih podataka jest ime slike i njena oznaka. Podatke o imenima i oznakama pohranio sam u liste za treniranje i testiranje neuronske mreže. Napokon na kraju sam liste s oznakama podataka za treniranje i testiranje pretvorio u tenzor koji ću kasnije koristiti pri provlačenju podataka kroz model. Prethodno opisano ponašanje nalazi se u sljedećem programskom odsječku:

```
import torch
import pandas
import numpy as np

df = pandas.read_csv('HAM10000_metadata.csv')
df = df.sample(frac=1)

# labels of different skin leisons (classification targets)
dx_values = ["akiec", "bcc", "bkl", "df", "mel", "nv", "vasc"]

train_image_names = []
train_labels = []
test_image_names = []
test_labels = []

number_of_images = len(df)
n_train = int(number_of_images * 0.80)

for i in range(number_of_images):
  row = df[i : i+1]

  # extract the values from the row
  lesion_id, image_id, dx, dx_type, age, sex, localization = row.values[0]

  # load the image and add it to the image names array
  name = "images/" + image_id + ".jpg"

  if i < n_train:
    train_image_names.append(name)
    train_labels.append(dx_values.index(dx))
  else:
    test_image_names.append(name)
    test_labels.append(dx_values.index(dx))

train_labels = torch.tensor(train_labels)
test_labels = torch.tensor(test_labels)
```

Što se tiče predobrade podataka, jedino što je bilo potrebno napraviti je prilikom učitavanja slike, smanjiti njene dimenzije budući da je početna dimenzija slike bila

600x450. Slike sam smanjio i po visini i po širini 3 puta, pa je tako dimenzija slike nakon smanjivanja bila 200x150. Nove dimenzije slike su, iako smanjene 3 puta, i dalje bile dovoljne veličine da model može naučiti klasificirati slike iz danih podataka.

4.3. Odabir i treniranje modela

4.3.1. Odabir modela

Model koji sam odlučio koristiti jest ResNet-50 mreža, jer je to prilično jak model, a već je ranije spomenuto da rezidualne mreže ne gube performanse povećavanjem dubine mreže. ResNet-50 je mreža koja koristi rezidualne blokove i sastoji se od 50 slojeva.

PyTorch nudi već gotove implementacije nekih rezidualnih neuronskih mreža, poput ResNet-18, ResNet-34, ResNet-50, ResNet-101 te ResNet-150 i iz tog sam se razloga odlučio koristiti njihovu implementaciju mreže. Mreža je bila predtrenirana na $ImageNet$ skupu podataka te sam ju ja dotrenirao na svojem skupu. Bilo je potrebno dodati jednu preinaku u mrežu prije nego li je ona spremna za uporabu. Naime, implementacija mreže je takva da je broj izlaznih značajki bio 1000, a za ovaj problem je bilo potrebno 7 izlaznih značajki, jer postoji 7 različitih bolesti koje slika može predstavljati. Kako bih riješio taj problem, maknuo sam posljednji potpuno povezani sloj, koji je imao 2048 ulaznih značajki te 1000 izlaznih, i zamijenio ga s potpuno povezanim slojem koji ima također 2048 ulaznih značajki, no broj izlaznih značajki je 7. Model i tenzore je zatim bilo potrebno izmijeniti tako da budu prikladni za evaluaciju na grafičkoj kartici. Budući da se program izvršavao na Google-ovom Colabu, dopušten je pristup njihovim grafičkim karticama. Postupak izmjene modela i tenzora da budu prikladni za evaluaciju na grafičkoj kartici, olakšan je $PyTorch$-evim paketom $torch.cuda$ koji omogućuje podršku za izvođenja na grafičkoj kartici. Postupak je izmjene je slijedio ovako:

1. dohvaćanje dostupnog $cuda$ uređaja $device = torch.device(\text{"}cuda : 0\text{"})$
2. prebacivanje modela na dostupni uređaj: $model.to(device)$
3. prebacivanje svih korištenih tenzora na dostupni uređaj: $inputs = tensor.to(device)$

4.3.2. Treniranje modela

Model je treniran na skupu slika za treniranje kojih je bilo oko 8000, a ostalih 2000 slika je služilo kao skup za evaluaciju modela. Veličina jedne minigrupe slika je bile 16, dok je broj epoha bio 30. PyTorch biblioteka nudi i mogućnost spremanja utreniranog modela što je bilo neophodno za kasniju evaluaciju na testnim podacima te kako ne bih morao svaki put iznova utreniravati 30 epoha modela. Trajanje prve epohe treniranja trajalo je bitno dulje od narednih epoha jer su se slike prvi puta učitavale s Google Diska (engl. *Google Drive*) u virtualni stroj na kojem se provodi učenje modela. Nakon što je prva epoha završila svaka iduća je trajala u prosjeku 4 do 5 minuta. Kao funkciju gubitka koristio sam softmax s unakrsnom entropijom (engl. *Cross-Entropy Loss*), a kao optimizator je korišten Adam s pretpostavljenim postavkama.

4.4. Eksperimentalni rezultati

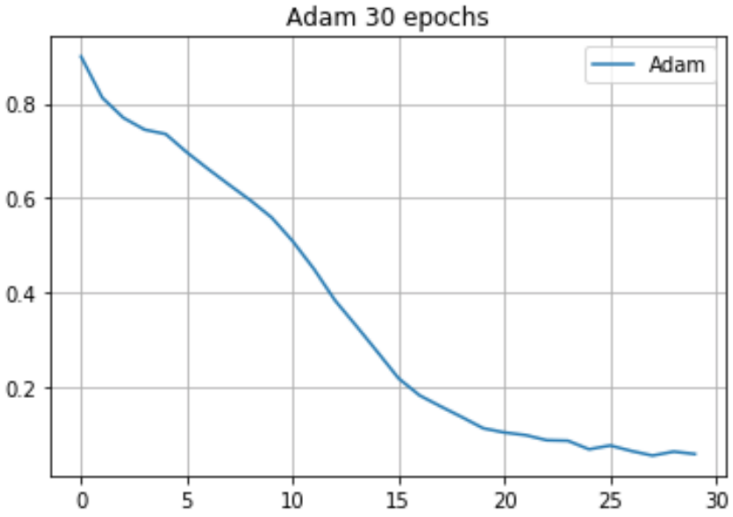
Treniranje se odvijalo kroz 30 epoha i nakon svake epohe bilježio sam iznos gubitka, kako bih mogao iscrtati dijagram promjene gubitka u odnosu na broj epoha, taj dijagram je prikazan u nastavku.

Cjelokupni program izvršavao se na Google-ovom servisu Colab, koji omogućava besplatno korištenje njihovih grafičkih kartica i time omogućavaju da se kompleksni modeli treniraju relativno brzo, no i dalje mnogo puta brže nego na centralnim procesnim jedinicama (CPU). U vrijeme pisanja ovog rada, Google je omogućio korištenje NVIDIA Tesla T4 grafičkih kartica. S prednosti mogućnosti korištenja grafičke kartice za treniranje neuronske mreže, model je veoma brzo počeo konvergirati, jer je trajanje epohe bilo u prosjeku 4 do 5 minuta, izuzev prve epohe koja je trajala znatno duže zbog inicijalnog učitavanja slika u virtualni stroj, te jednom kada su bile učitane, sve je teklo mnogo brže.

Nakon evaluiranja modela na skupu podataka za ispitivanje, model je ostvario točnost od 89.97%.

4.4.1. Matrica zabune (engl. *Confusion matrix*)

Matrica zabune govori nam koliko je model dobro klasificirao podatke iz pojedine razrede. Čak štoviše, pokazuje u koje je sve razrede model pogrešno(ili točno) svrstao podatak iz svakog pojedinog razreda. Svaki redak matrice predstavlja točan razred podatka, dok svaki stupac predstavlja predviđanje modela za podatak. Na taj način



Slika 4.1: Dijagram iznosa gubitka po epohama.

možemo uvidjeti za svaki razred gdje je model najviše griješio. Normalizirana i ne normalizirana matrica zabune za testni skup od 2003 podataka prikazane su u nastavku.

Model dobro raspoznaje većinu kožnih nepravilnosti međutim ponekad ima problema pri razlikovanju Bowenove bolesti s karcinomom bazalnih stanica i benignim nepravilnostima sličnih keratozi. Također slična je situacija i sa dermatofibromom kojeg model ponekad zamijeni s Bowenovom bolesti ili karcinomom bazalnih stanica.

	Bowen's disease	Basal cell carcinoma	Benign keratosis-like lesions	Dermatofibroma	Melanoma	Melanocytic nevus	Vascular lesions
Bowen's disease	46	8	6	0	5	1	0
Basal cell carcinoma	3	93	6	0	4	3	3
Benign keratosis-like lesions	4	2	141	0	12	29	0
Dermatofibroma	2	2	0	14	0	1	0
Melanoma	4	0	10	0	168	31	0
Melanocytic nevus	1	9	19	2	32	1305	2
Vascular lesions	0	2	0	0	2	1	30

Slika 4.2: Matrica zabune bez normalizacije.

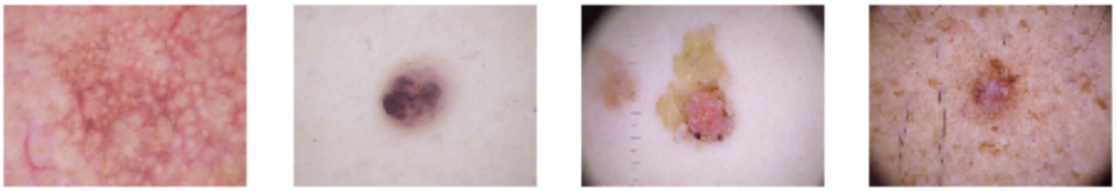
	Bowen's disease	Basal cell carcinoma	Benign keratosis-like lesions	Dermatofibroma	Melanoma	Melanocytic nevus	Vascular lesions
Bowen's disease	0,70	0,12	0,09	0	0,08	0,02	0
Basal cell carcinoma	0,03	0,83	0,05	0	0,04	0,03	0,03
Benign keratosis-like lesions	0,02	0,01	0,75	0	0,06	0,15	0
Dermatofibroma	0,11	0,11	0	0,74	0,00	0,05	0
Melanoma	0,02	0	0,05	0	0,79	0,15	0
Melanocytic nevus	0	0,01	0,01	0	0,02	0,95	0
Vascular lesions	0,00	0,06	0,00	0,00	0,06	0,03	0,86

Slika 4.3: Normalizirana matrica zabune.

Prilikom evaluiranja modela uzeo sam nekoliko primjera dobrih i loših klasifikacija, koje sam ranije spomenuo da se događaju kod Bowenove bolesti i dermatofibroma.

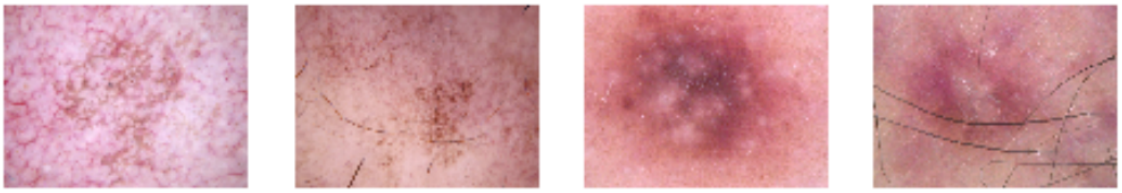
Na slici 4.4 prikazane su 4 bolesti kod kojih je model griješio, a na slici 4.5 prikazane su netočno klasificirane bolesti. Slika 4.4 ovdje služi čitatelju kao referenca izgleda pojedine bolesti. Slika 4.5 s lijeva na desno prikazuje:

1. točan razred: Bowenova bolest, a predviđen je karcinom bazalnih stanica
2. točan razred: Bowenova bolest, a predviđena je benigna nepravilnost slična keratozi
3. točan razred: dermatofibrom, a predviđena je Bowenova bolest
4. točan razred: dermatofibrom, a predviđen je karcinom bazalnih stanica



Slika 4.4: S lijeva na desno (točan razred - izlaz iz modela).

Bowenova bolest - Bowenova bolest, karcinom bazalnih stanica - karcinom bazalnih stanica, benigna nepravilnost slična keratozi - benigna nepravilnost slična keratozi, dermatofibrom - dermatofibrom.



Slika 4.5: Netočno klasificirane bolesti, s lijeva na desno (točan razred - izlaz iz modela).

Bowenova bolest - karcinom bazalnih stanica, Bowenova bolest - benigna nepravilnost slična keratozi, dermatofibrom - benigna nepravilnost slična keratozi, dermatofibrom - karcinom bazalnih stanica.

5. Zaključak

Kroz ovaj rad upoznao sam se s radnim okruženjem PyTorch i implementirao rješenje za problem klasifikacije slika kožnih lezija prikupljenih u svrhe vježbe i povećanja zanimanja mladih ljudi za uporabu metoda strojnog učenja u medicini.

Koristeći ResNet-50 model postigao sam odlične rezultate klasifikacije slika i to je pokazatelj da su rezidualne mreže veoma efikasne u rješavanju tog problema te da za razliku od klasičnih konvolucijskih mreža, ne gube performanse s povećanjem dubine mreže.

Microsoftov istraživački tim je otvorio nova vrata u području računalnog vida, dizajniranjem rezidualnih neuronskih mreža i pokazao da se primjenom jednostavnog koncepta prečaca može znatno poboljšati efikasnost i performansa konvolucijske neuronske mreže.

Iako je na ovom skupu podataka rezidualna mreža ResNet-50 pokazala koliko dobre rezultate može dati, rezultati uvelike ovise i o samim podacima. Kada govorimo o podacima, iako to često uzimamo zdravo za gotovo, bitno je spomenuti da prikupljeni uzorci slika pojedinih bolesti moraju biti reprezentativni i dobro opisivati populaciju koja ima tu bolest. Bez te pretpostavke ovaj model ne bi mogao dati dobre rezultate na slikama koje nisu iz ovog skupa podataka, a tiču se navedenih bolesti.

LITERATURA

Neural netowrks architectures. URL https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html. Datum pristupa 30.05.2019.

Convolutional neural networks. URL http://cs231n.github.io/convolutional-networks/. Datum pristupa 30.05.2019.

Seborrheic keratosis. URL https://www.healthline.com/health/seborrheic-keratosis. Datum pristupa 31.05.2019.

Računalni vid, Rujan 2013. URL https://hr.wikipedia.org/wiki/RaÄ■unalni_vid. Datum pristupa 30.05.2019.

An intuivive explanation of convolutional neural networks, Kolovoz 2016. URL https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. Datum pristupa 30.05.2019.

Što su dermatofibromi, Prosinac 2016. URL https://www.zdravobudi.hr/clanak/795/sto-su-dermatofibromi. Datum pristupa 31.05.2019.

Supervised and unsupervised machine learning algorithms, Ožujak 2016. URL https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. Datum pristupa 12.06.2019.

Unsupervised learning: Dimensionality reduction, Travanj 2017. URL https://towardsdatascience.com/unsupervised-learning-dimensionality-reduction-ddb4d55e0757. Datum pristupa 12.06.2019.

Machine learning for humans, Kolovoz 2017. URL https://medium.com/machine-learning-for-humans/

why-machine-learning-matters-6164faf1df12. Datum pristupa 12.06.2019.

Common loss functions in machine learning, Rujan 2018. URL https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23. Datum pristupa 12.06.2019.

Activation function, Svibanj 2019. URL https://en.wikipedia.org/wiki/Activation_function. Datum pristupa 30.05.2019.

Artificial neuron, Veljača 2019. URL https://en.wikipedia.org/wiki/Artificial_neuron. Datum pristupa 30.05.2019.

Basal-cell carcinoma, Lipanj 2019. URL https://en.wikipedia.org/wiki/Basal-cell_carcinoma. Datum pristupa 31.05.2019.

Bowen's disease, Svibanj 2019. URL https://www.nhs.uk/conditions/bowens-disease/. Datum pristupa 31.05.2019.

Convolutional neural network, Svibanj 2019. URL https://en.wikipedia.org/wiki/Convolutional_neural_network. Datum pristupa 30.05.2019.

Melanocytic nevus, Svibanj 2019a. URL https://en.wikipedia.org/wiki/Melanocytic_nevus. Datum pristupa 31.05.2019.

Melanoma, Lipanj 2019b. URL https://en.wikipedia.org/wiki/Melanoma. Datum pristupa 31.05.2019.

Neuron, Lipanj 2019. URL https://en.wikipedia.org/wiki/Neuron. Datum pristupa 01.06.2019.

Rectifier (neural networks), Svibanj 2019. URL https://en.wikipedia.org/wiki/Rectifier_(neural_networks). Datum pristupa 30.05.2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Identity mappings in deep residual networks. *Lecture Notes in Computer Science*, stranica 630–645, 2016. ISSN 1611-3349. doi: 10.1007/978-3-319-46493-0_38. URL http://dx.doi.org/10.1007/978-3-319-46493-0_38.

Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. 2015. Datum pristupa 30.05.2019.

Sagar Sharma. Activation function neural networks, Rujan 2017. URL `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`. Datum pristupa 30.05.2019.

Andreas Veit, Michael Wilber, i Serge Belongie. Residual networks behave like ensembles of relatively shallow networks, 2016.

Geoffrey E. Hinton Vinod Nair. Rectified Linear Units Improve Restricted Boltzmann Machines. *International Conference on Machine Learning (ICML-10)*, 2010.

Bing Xu, Naiyan Wang, Tianqi Chen, i Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

Klasifikacija medicinskih slika konvolucijskim modelima

Sažetak

Klasifikacija slika je jedan od još neriješenih problema računalnog vida, no iz godine u godinu se vidi napredak u tom području. U ovom radu opisane su konvolucijske neuronske mreže te njihova primjena u klasifikaciji slika. Dodatno opisane su i rezidualne neuronske mreže koje su omogućile velik pomak u području računalnog vida. U okviru ovog rada koristio sam implementaciju rezidualne mreže ResNet-50 za problem klasifikacije slika kožnih nepravilnosti na skupu podataka HAM:10000. Dobiveni rezultati su prikazani i diskutirani na kraju rada.

Ključne riječi: duboko učenje, računalni vid, klasifikacija slika, medicinske slike, konvolucijske neuronske mreže, Pytorch

Medical Image Classification With Convolutional Models

Abstract

Image classification is still one of the unsolved computer vision problems, but there is a clear advance in that area from year to year. This paper describes convolutional neural networks and their application in classification problem. In addition, there is a thorough description of residual neural networks, which enabled a major leap forward in computer vision. For the purposes of this work I used ResNet-50 residual network to solve the problem of skin lesion image classification from the dataset HAM:10000. The results of the experiment are displayed and commented at the end of the paper.

Keywords: deep learning, computer vision, image classification, medical images, convolutional nerual networks, Pytorch