SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 11

SEMANTIČKA SEGMENTACIJA NOVOTVORINA U SLIKAMA MAGNETSKE REZONANCIJE

Lovro Rabuzin

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 11

SEMANTIČKA SEGMENTACIJA NOVOTVORINA U SLIKAMA MAGNETSKE REZONANCIJE

Lovro Rabuzin

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 11

Pristupnik:	**Lovro Rabuzin (0036513931)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Semantička segmentacija novotvorina u slikama magnetske rezonancije**

Opis zadatka:

Semantička segmentacija važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate postižu konvolucijski modeli s ljestvičastim naduzorkovanjem. Ovaj rad razmatra primjenu tog postupka za segmentaciju novotvorina u medicinskim slikama koje su dobivene magnetskom rezonancijom. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće pristupe za klasifikaciju slike. Odabrati slobodno dostupni skup slika te oblikovati podskupove za učenje, validaciju i testiranje. Predložiti prikladnu arhitekturu dubokog modela za semantičku segmentaciju novotvorina. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele te prikazati i ocijeniti postignutu točnost. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 11. lipnja 2021.

Zahvaljujem mentoru prof. dr. sc. Siniši Šegviću na svoj pruženoj pomoći i izdvojenom vremenu.

Hvala i mojim prijateljima i obitelji na moralnoj podršci i savjetima.

SADRŽAJ

1. Uvod

Računalni je vid područje umjetne inteligencije koje se bavi obradom informacija iz slika na način sličan ljudima. Inovacije u dubokom učenju i neuronskim mrežama omogućile su značajne napretke u području računalnog vida.

Semantička segmentacija jedan je od ključnih problema kojima se bavi računalni vid. Radi se o pridjeljivanju oznake odgovarajuće klase svakom pikselu slike. Semantička segmentacija koristi se u mnoštvo svrha, od kojih su najpopularnije autonomna vožnja, industrijska inspekcija proizvoda, prepoznavanje objekata na satelitskim snimkama te analiza medicinskih snimki.

U ovom radu bavit ću se upravo analizom medicinskih snimki, preciznije semantičkom segmentacijom tumora i njihovih dijelova u slikama magnetske rezonancije. Segmentacija tumora prvi je korak u liječenju tumora nakon dijagnoze. Većina tumora liječi se radioterapijom koja često oštećuje i okolno zdravo tkivo te je zbog toga bitno točno odrediti gdje se nalaze tumorske stanice, a gdje normalno tkivo. Segmentacijom tumora obično se bave radiolozi i onkolozi. Takva ručna segmentacija traje satima i mora se provoditi više puta tijekom procesa liječenja, što predstavlja veliki teret za zdravstveni sustav.

Automatizirana segmentacija tumora nudi se kao jedno od mogućih rješenja za taj problem. Konvolucijske neuronske mreže pokazale su se kao dobro rješenje za problem semantičke segmentacije pa ću ih i ja koristiti u sklopu ovog rada.

U prva dva poglavlja opisat ću osnovne koncepte dubokog učenja i semantičke segmentacije. U sljedeća tri poglavlja opisat ću korišteni skup podataka i arhitekture mreža. U posljednja tri poglavlja predstavit ću detalje svoje implementacije i dobivene rezultate te prokomentirati rezultate i ponuditi moguća poboljšanja.

2. Duboko Učenje

2.1. Umjetne neuronske mreže

Kao ljudi, sposobni smo svake sekunde izvući čitavi niz informacija iz podražaja koje dobivamo osjetilom vida. Tu sposobnost obrade informacija iz okoliša omogućuje nam mozak. Kroz istraživanja u područjima znanosti poput neurofiziologije i kognitivnih znanosti spoznali smo da je osnovna gradivna jedinica mozga neuron. Također znamo da se mozak sastoji od mnoštva neurona koji su međusobno povezani i tako omogućuju paralelnu obradu podataka.

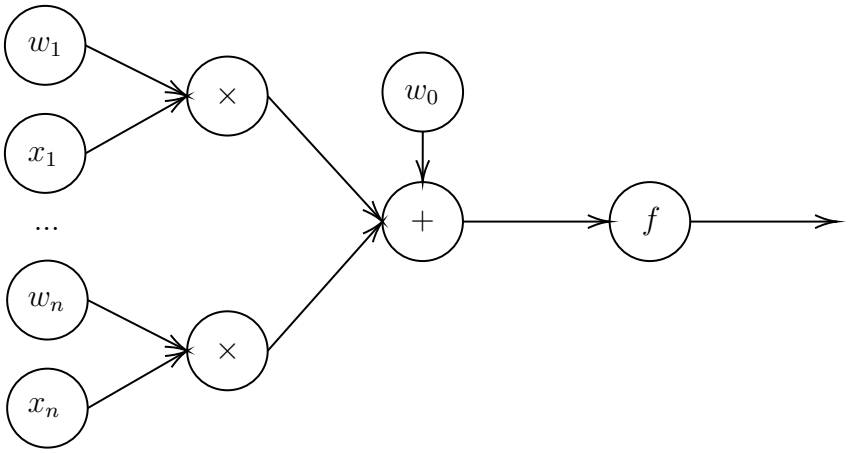
Umjetne neuronske mreže inspirirane su opisanim biološkim procesom te se također sastoje od mnoštva manjih gradivnih jedinica koje nazivamo neuronima. Unatoč biološkoj inspiraciji iza umjetnih neurona, umjetne neuronske mreže ne modeliraju stvarne biološke procese unutar mozga. Neuronske mreže zapravo predstavljaju nelinearne funkcije koje preslikavaju ulazne vrijednosti na izlazne. Jedino ograničenje na funkcije koje modeliramo neuronskim mrežama je da moraju biti jednom diferencijabilne gotovo svugdje. Obično se radi o kompoziciji jednostavnijih funkcija te ih zato izračunavamo računskim grafovima koje gradimo iz izvornog koda u višem programskom jeziku.

Pregled umjetnih neuronskih mreža u nastavku načinjen je prema [23], [25], [4] i predavanjima iz kolegija Duboko učenje na Fakultetu elektrotehnike i računarstva koja su dostupna na [24].

2.1.1. Neuron

Inspirirani načinom funkcioniranja biološkog neurona, dvojica znanstvenika Warren McCulloch i Walter Pitts 1943. godine u [12] predstavili su model umjetnog neurona prikazan računskim grafom na slici 2.1.

Pobudu umjetnog neurona čine ulazi $x_1, x_2, ..., x_n$ koji predstavljaju izlaze prethodnih neurona ili određene atribute ulaznih podataka. Svaka pobuda x_i množi se



Slika 2.1: Umjetni neuron prikazan kao računski graf. Umjetni neuron prikazan ovdje predstavlja osnovni neuron koji su Warren McCulloch i Walter Pitts predložili 1943. godine. Ovaj neuron sastoji se od skalarnog produkta težina neurona $\boldsymbol{w}$ i ulaza $\boldsymbol{x}$ te nelinearne aktivacijske funkcije f. Osnovne gradivne jedinice modernih neuronskih mreža često uključuju i druge vrste čvorova, kao što je normalizacija nad grupom. Slika je napravljena po uzoru na [4].

pripadnom težinom w_i koja određuje utjecaj pobude na izlaz neurona. Tijelo umjetnog neurona zbraja dobivene umnoške i prag w_0. Na dobivenu sumu primjenjuje se prijenosna funkcija f da bi se dobio izlaz neurona o, kao što je opisano izrazom (2.1):

$$o = f\left(\sum_{i=1}^{n} x_i \cdot w_i + w_0\right) \tag{2.1}$$

Ovakav umjetni neuron predstavlja temeljnu gradivnu jedinicu neuronske mreže. Neuroni modeliraju jednostavne funkcije koje slažemo u računski graf, to jest neuronsku mrežu. Takvim računskim grafom opisujemo kako komponiramo osnovne funkcije da bismo dobili složenu funkciju koja dobro opisuje naše podatke. U modernim neuronskim mrežama, neuroni sadrže i druge operacije osim skalarnog produkta i nelinearnosti. Neke od takvih operacija su normalizacija nad grupom ili deformabilna konvolucija.

2.1.2. Prijenosne funkcije

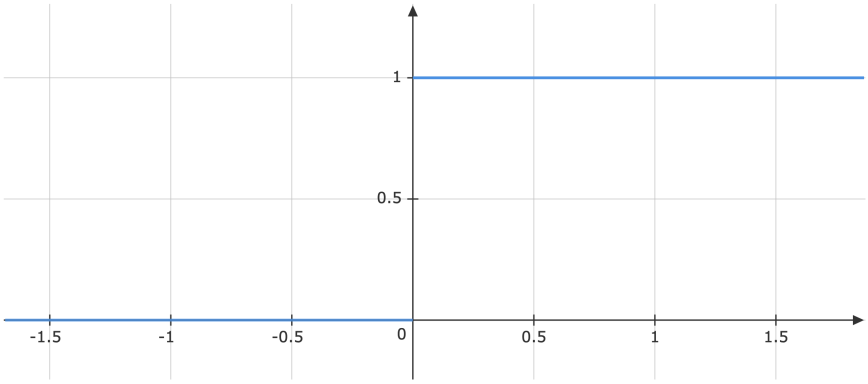
Kada bismo koristili linearne prijenosne funkcije ili samo funkciju identiteta, ulančavanjem više slojeva neurona ne bismo mogli rješiti išta više od trivijalnih problema. Kako je kompozicija linearnih funkcija također linearna funkcija, čitavu mrežu takvih neurona mogli bismo zamijeniti samo jednim neuronom s odgovarajućim težinama i pragom. Zbog toga za rješavanje kompleksnijih problema koristimo nelinearne prije-

nosne funkcije.

Funkcija skoka

McCulloch i Pitts koristili su funkciju skoka opisanu izrazom (2.2) kao prijenosnu funkciju u svojem modelu TLU-perceptrona (engl. *Threshold Logic Unit*).

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0. \end{cases} \tag{2.2}$$



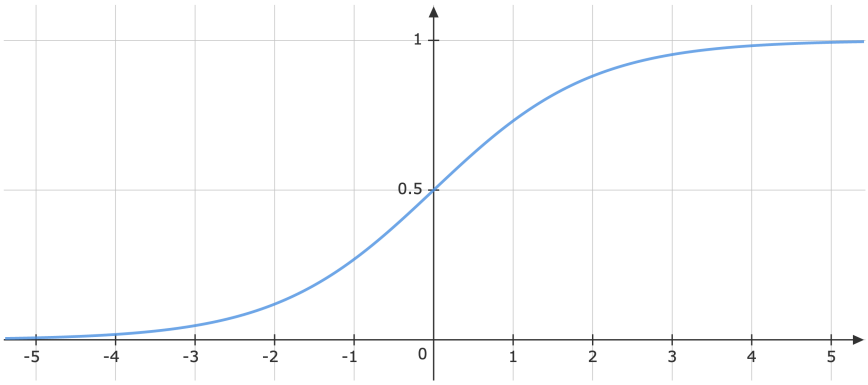
Slika 2.2: Funkcija skoka

Današnje neuronske mreže uče algoritmom propagacije pogreške unatrag (engl. *backpropagation*) koji računa gradijente izlaza neurona. Kako je derivacija funkcije skoka 0 svugdje osim u točki $x = 0$, gdje je prekid, nije prikladna za korištenje s algoritmom propagacije greške unatrag pa se danas uglavnom ne koristi.

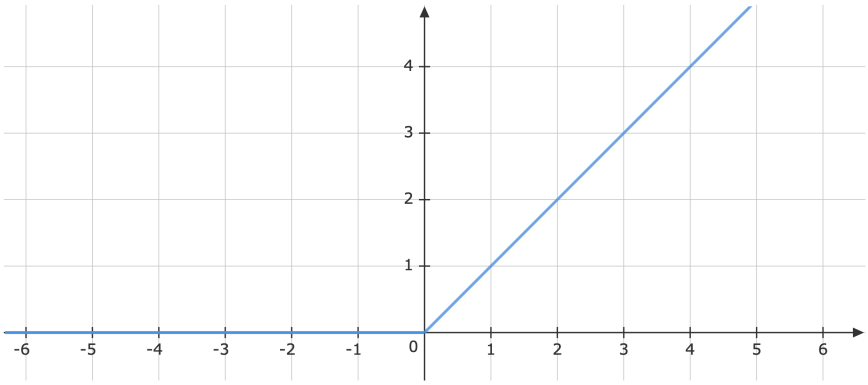
Sigmoidalna funkcija

Sigmoidalnu funkciju možemo smatrati poopćenjem funkcije skoka. Sigmoidalna funkcija opisana izrazom (2.3) postupno raste od 0 do 1. Za razliku od funkcije skoka, sigmoida ima derivacije koje nisu 0 pa je prikladnija za korištenje s algoritmima učenja koji koriste gradijente izlaza neurona.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$



Slika 2.3: Sigmoida



Slika 2.4: Zglobnica

Zglobnica

Zglobnica (engl. *rectified linear unit*, kratica *relu*) nelinearna je prijenosna funkcija opisana izrazom (2.4) koja pozitivne vrijednosti propušta, a negativne vrijednosti preslikava u nulu.

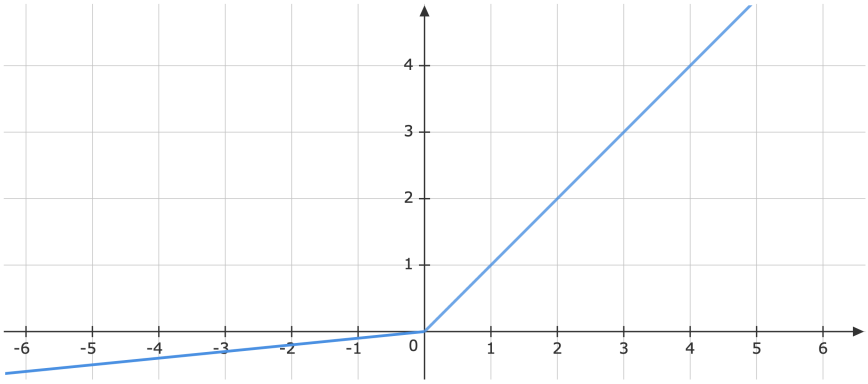
$$f(x) = \max(0, x) \tag{2.4}$$

Zglobnica je danas najšire korištena prijenosna funkcija u dubokim umjetnim neuronskim mrežama. Unatoč tome, i zglobnica ima svoj problem. Kako je derivacija zglobnice 1 za pozitivne ulaze, a 0 za negativne ulaze, može se dogoditi da neuron prestane učiti kada dobivamo negativne ulaze, to jest umire.

Propusna zglobnica

Propusna zglobnica (engl. *leaky rectified linear unit*, kratica *leaky relu*) opisana izrazom (2.5) modifikacija je zglobnice koja rješava problem umirućih neurona.

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha \cdot x, & x \leq 0. \end{cases} \tag{2.5}$$



Slika 2.5: Propusna zglobnica s $\alpha = 0.1$

Parametar α obično je mali pozitivan broj. Derivacija propusne zglobnice je 1 za pozitivne ulaze i α za negativne ulaze. Kako je derivacija uvijek broj različit od nule, neuroni koji koriste propusnu zglobnicu kao prijenosnu funkciju uvijek mogu učiti kada koristimo postupke učenja koji koriste derivacije.

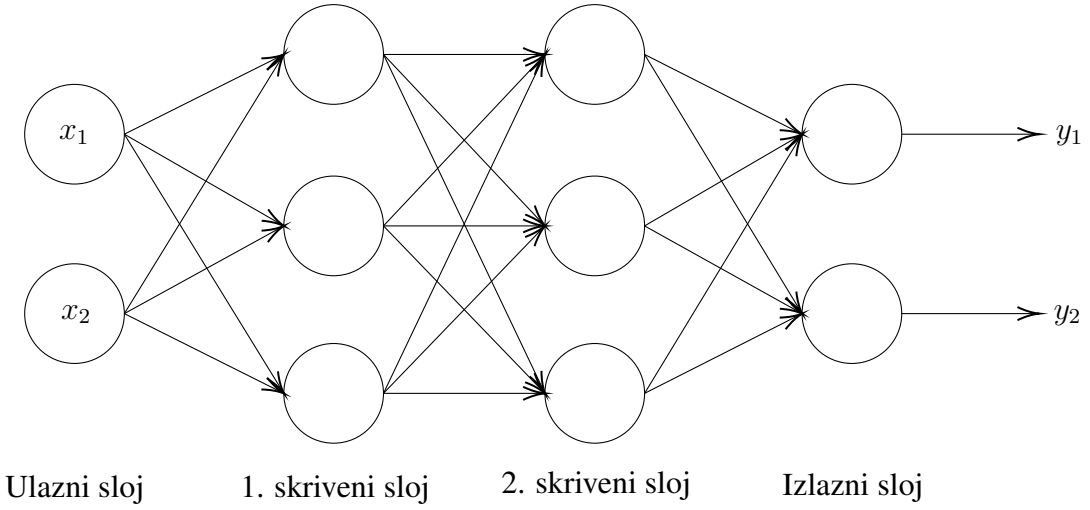
2.1.3. Arhitektura mreže

Neuronske mreže sastoje se od povezanih neurona koji su međusobno podijeljeni u slojeve. Prvi sloj, kojemu prosljeđujemo podatke za učenje, naziva se ulazni sloj. Na ulazni sloj može se nadovezivati više slojeva. Posljednji sloj, koji daje izlaz iz mreže naziva se izlazni sloj. Slojevi između ulaznog i izlaznog nazivaju se skrivenim slojevima. Ako su slojevi potpuno povezani, svaki neuron iz sloja i dobiva pobudu iz svih neurona iz sloja $i - 1$.

Osnovno svojstvo neuronske mreže je kapacitet. On opisuje sposobnost prilagođavanja podacima te je proporcionalan broju stupnjeva slobode modela. Modeli malog kapaciteta skloni su podnaučenosti, to jest, ne postoji skup parametara modela koji mogu adekvatno objasniti skup za učenje. Nasuprot tome, modeli velikog kapaciteta skloni su prenaučenosti, to jest model počinje učiti i šum u podacima.

Unaprijedne neuronske mreže

Ako se niti jedan izlaz iz sloja neuronske mreže ne vraća kao ulaz nekog od slojeva, to jest nema petlji, mrežu nazivamo unaprijednom neuronskom mrežom (engl. *feed-forward neural network*). Takve mreže obično predstavljamo usmjerenim acikličkim računskim grafovima. Unaprijedne neuronske mreže služe kao baza za mnoge primjene neuronskih mreža za rješavanje problema iz stvarnog svijeta. Konvolucijske neuronske mreže koje se koriste za prepoznavanje predmeta u slikama, čime se i ovaj rad bavi, također su unaprijedne neuronske mreže. Na slici 2.6 prikazan je primjer unaprijedne potpuno povezane neuronske mreže.



Slika 2.6: Unaprijedna potpuno povezana neuronska mreža. Ovdje svaki čvor predstavlja skalarni produkt ulaza i težina te nelinearnu prijenosnu funkciju. Slika je napravljena po uzoru na [23].

Unaprijedne neuronske mreže mogu se predstaviti kompozicijom jednostavnijih funkcija, gdje svaka funkcija modelira jednu nelinearnu aktivaciju afinog preslikavanja.

2.2. Učenje neuronske mreže

Neuronske mreže sastoje se od više slojeva međusobno povezanih neurona. Svaki od tih neurona ima svoje težine i pragove. Da bismo dobili željeni izlaz, te parametre potrebno je podesiti tako da mreža najbolje opisuje podatke koje obrađuje. Takvo podešavanje parametara neurona naziva se učenje neuronske mreže. U ovom radu bavit ću se nadziranim učenjem neuronskih mreža (engl. *supervised learning*). To znači da znamo očekivane izlaze neuronske mreže. Tijekom učenja mreže, parametre

podešavamo tako da dobivamo čim točnije izlaze za ulaze kojima znamo očekivani izlaz i nadamo se da će mreža nakon učenja dobro predviđati vrijednosti za ulaze koje nikad nije vidjela. Ta sposobnost mreže naziva se generalizacija.

Do savršenih parametara mreže ne možemo doći iscrpnom pretragom. Parametara je jednostavno previše i mogu poprimiti previše vrijednosti da bismo tako u razumnom vremenu pronašli skup parametara koji savršeno opisuje naše podatke. Zato učenje neuronske mreže rješavamo kao optimizacijski problem za koji postoji algoritam koji pretražuje prostor mogućih rješenja za koja dobivamo dobre rezultate.

2.2.1. Funkcija gubitka

Funkcija gubitka ciljna je funkcija u problemu optimizacije parametara neuronske mreže. Ona opisuje koliko se izlazi neuronske mreže razlikuju od očekivanih izlaza te je, kao takvu, želimo minimizirati. Različite funkcije gubitka prikladne su za različite primjene.

Unakrsna entropija

Za klasifikaciju, čime ću se i baviti u ostatku ovog rada, kao funkciju pogreške prikladno je koristiti unakrsnu entropiju (engl. *cross-entropy loss*), opisanu izrazom (2.6):

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} t_{c_i} \cdot \ln p_{c_i} \tag{2.6}$$

Pritom, n je broj podataka koje imamo u skupu, t_{c_i} predstavlja oznaku pripada li ulaz i razredu c i poprima vrijednosti 0 ili 1, a p_{c_i} označava procijenjenu vjerojatnost pripada li ulaz i razredu c. Vrijednost t_{c_i} bit će 1 samo za razred kojem svaki primjer zbilja pripada, tako da unakrsnu entropiju možemo ekvivalentno zapisati kao u jednadžbi (2.7):

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n} \ln p_{c_i} \tag{2.7}$$

2.2.2. Gradijentni spust

Da bismo našli minimum funkcije gubitka s obzirom na parametre modela, koristimo gradijentni spust. Ako možemo izračunati gradijent funkcije gubitka, tada znamo da parametre moramo mijenjati u smjeru negativnog gradijenta kako bismo se približili minimumu.

Funkcija gubitka L skalarna je funkcija više varijabli, što znači da će njezin gradijent biti jednak transponiranoj Jakobijevoj matrici.

$$\nabla L(\boldsymbol{x}) = \frac{dL(\boldsymbol{x})}{d\boldsymbol{x}}^{\intercal} \tag{2.8}$$

Ponašanje funkcije gubitka L oko točke $\boldsymbol{x}$ možemo aproksimirati Taylorovim razvojem prvog reda.

$$L(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx L(\boldsymbol{x}) + \frac{dL(\boldsymbol{x})}{d\boldsymbol{x}}\Delta x = L(\boldsymbol{x}) + \nabla L(\boldsymbol{x})^{\intercal}\Delta x \tag{2.9}$$

Kao što sam ranije spomenuo, želimo se kretati u smjeru negativnog gradijenta pa kao pomak u $\boldsymbol{x}$ uvrštavamo kretanje u negativnom smjeru gradijenta.

$$\Delta\boldsymbol{x} = -\eta\nabla L(\boldsymbol{x}) \tag{2.10}$$

Ovdje je η hiperparametar koji zovemo korak učenja te poprima male pozitivne vrijednosti.

Izraz (2.9) sada poprima oblik dan izrazom (2.11).

$$L(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx L(\boldsymbol{x}) - \eta\nabla L(\boldsymbol{x})^{\intercal}\nabla L(\boldsymbol{x}) = L(\boldsymbol{x}) - \eta \left\|\nabla L(\boldsymbol{x})\right\|^2 \tag{2.11}$$

Kako uvijek vrijedi $\left\|\nabla L(\boldsymbol{x})\right\|^2 \geq 0$, vidimo da se pomicanjem u smjeru negativnog gradijenta uvijek smanjuje iznos funkcije gubitka.

Dakle, da bismo smanjili iznos funkcije gubitka, parametrima modela u svakom koraku učenja pridružujemo nove vrijednosti koje se računaju prema izrazu (2.12):

$$w_i \leftarrow w_i - \eta\frac{\delta L}{\delta w_i} \tag{2.12}$$

U praksi, gradijent ne računamo za čitavi skup podataka jer zahtijeva previše računalnih resursa. Umjesto toga, podatke nasumično dijelimo u manje grupe (engl. *batch*) i nakon svake grupe ažuriramo parametre modela dobivenom procjenom gradijenta. Taj postupak naziva se stohastički gradijentni spust (engl. *stochastic gradient descent*, kratica SGD).

2.2.3. Propagacija pogreške unatrag

Kada koristimo unaprijednu neuronsku mrežu, mreža prihvaća ulaz i generira izlaz. Ulaz daje inicijalnu informaciju koja teče unaprijed kroz skrivene slojeve do zadnjeg sloja koji daje izlaz. Na temelju izlaza mreže procjenjujemo iznos funkcije gubitka.

Algoritam propagacije pogreške unatrag dopušta informaciji da teče unatrag, od funkcije gubitka kroz mrežu kako bi se izračunao gradijent. Algoritam propagacije pogreške koristimo jer nam daje način računanja gradijenata koji ne zauzima puno računalnih resursa

Propagacijom pogreške unatrag možemo izračunati gradijent proizvoljne funkcije po proizvoljnom skupu parametara te funkcije, no u dubokom učenju koristi se za računanje gradijenta funkcije gubitka po parametrima neuronske mreže $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$.

Zbog pravila ulančavanja (engl. *chain rule*), parcijalne derivacije funkcije greške iz sloja ispod mogu se računati uz pomoć parcijalnih derivacija iz sloja iznad. Tako izbjegavamo potrebu za računanjem parcijalnih derivacija u svakom sloju iz početka jer koristimo informacije iz sloja iznad.

2.2.4. Regularizacija

S modelom dovoljnog kapaciteta, možemo ostvariti proizvoljnu točnost na skupu za učenje, no postoji opasnost da model prevelikog kapaciteta neće moći dobro generalizirati jer je počeo učiti šum u našem skupu podataka, to jest počeo je učiti primjere za učenje "na pamet". U takvom slučaju rekli bismo da je model prenaučen (engl. *overfitted*). Budući da želimo da naš model dobro generalizira, pokušavamo izbjeći prenaučenost. Jedan je od načina izbjegavanja prenaučenosti imati dovoljno velik skup podataka, no to često nije ostvarivo zbog cijene nabavljanja dodatnih podataka. Drugi je način na koji možemo spriječiti prenaučenost regularizacija. Regularizacija se općenito odnosi na bilo koji postupak koji poboljšava generalizaciju bez da se nužno smanji pogreška na skupu za učenje. Tehnike regularizacije uzrokuju preferencu prema jednostavnijem modelu ili služe kao izraz nekog početnog znanja o problemu.

Regularizacija normom vektora parametara

Regularizacija normom vektora parametara jedan je od starijih pristupa regularizaciji koji se koristio u linearnim modelima kao što su linearna i logistička regresija.

Regularizacija normom vektora parametara kažnjava veću kompleksnost modela dodajući normu vektora parametara na funkciju gubitka L.

$$\tilde{L} = L + \alpha\Omega(\boldsymbol{\theta}) \tag{2.13}$$

U jednadžbi (2.13), $\alpha \in [0, \infty)$ je hiperparametar koji određuje relativni doprinos norme Ω. Kada naš algoritam minimizira funkciju $\tilde{L}$, smanjuje originalnu funkciju

gubitka L, ali i neku mjeru veličine parametara $\boldsymbol{\theta}$. Tipično normu vektora parametara računamo samo na težinama $\boldsymbol{w}$, ne na pragovima. To radimo jer za pragove treba manje podataka da bi se ispravno naučili budući da utječu samo na jednu varijablu dok težine upravljaju interakcijama dviju varijabli.

Jedan od popularnih odabira funkcije Ω je L^2 norma parametara koja se još zove *weight decay*. L^2 regularizacija opisana je izrazom (2.14):

$$\tilde{L} = L + \alpha\frac{1}{2}\|\boldsymbol{w}\|_2^2 = L + \frac{\alpha}{2}\boldsymbol{w}^\intercal\boldsymbol{w} \tag{2.14}$$

L^2 regularizacija tjera parametre koji ne doprinose značajno smanjenju funkcije gubitka prema 0, dok parametri koji značajno doprinose smanjenju funkcije gubitka ostaju relativno netaknuti.

Još jedan, manje popularan, oblik regularizacije normom vektora parametara je L^1 regularizacija dana izrazom (2.15):

$$\tilde{L} = L + \alpha\|\boldsymbol{w}\|_1 = L + \alpha\sum_i |w_i| \tag{2.15}$$

L^1 regularizacija također vodi manje bitne parametre prema 0, to jest istovremeno uči model i obavlja eliminaciju značajki.

Rano zaustavljanje

Kada radimo s modelima koji imaju velik broj parametara, primjećujemo da se gubitak na skupu za treniranje smanjuje s vremenom, no gubitak na skupu za validaciju u početku pada, ali s vremenom počinje rasti. Dakle, model s najmanjom greškom na skupu za validaciju i time vjerojatno manjom greškom na skupu za testiranje možemo dobiti vraćanjem parametara na onu vrijednost koja je bila kada je model davao najmanju grešku na skupu za validaciju. Ova strategija naziva se rano zaustavljanje i to je najčešći oblik regularizacije kod dubokih modela jer njihovo učenje može jako dugo trajati, a ovdje ne treba optimirati nikakve parametre. Rano zaustavljanje obično se implementira tako da se učenje zaustavlja nakon što se greška na validacijskom skupu nije smanjila ispod najniže zabilježene vrijednosti u unaprijed definiranom broju epoha učenja.

Uvećanje podataka

Najbolji način na koji možemo povećati generalizacijsku sposobnost modela je dati modelu više podataka. Naravno, u praksi na raspolaganju imamo ograničenu količinu

podataka. Tom problemu možemo doskočiti generiranjem umjetnih podataka tijekom učenja modela. Taj pristup regularizaciji pokazao se osobito učinkovitim za raspoznavanje slika jer same sadrže mnogo faktora varijacije koje lako možemo simulirati. Slučajne rotacije i translacije mogu uvelike poboljšati generalizacijsku sposobnost modela. Pritom je bitno ne primjenjivati transformacije koje bi promijenile klasu podataka. Primjerice, za klasičan problem prepoznavanja rukom pisanih znamenki, zrcaljenja i rotacije za $180°$ nisu prikladne jer bi mogle promijeniti očekivanu oznaku za sliku. Ako sliku znamenke 6 rotiramo za $180°$, radit će se o slici znamenke 9.

2.2.5. Optimizacijske metode

Optimizacija u neuronskim mrežama odnosi se na pronalaženje parametara $\boldsymbol{\theta}$ koji smanjuju funkciju gubitka $L(\boldsymbol{\theta})$. Ovo se razlikuje od tradicionalnih optimizacijskih algoritama jer se radi o indirektnoj optimizaciji. U tradicionalnim optimizacijskim algoritmima, optimiramo izravno mjeru P. U neuronskim mrežama optimiramo funkciju gubitka $L(\boldsymbol{\theta})$ i nadamo se da ćemo time neizravno optimirati i mjeru P, to jest grešku modela.

Funkcije gubitka kod neuronskih mreža nisu konveksne, što znači da imaju mnoštvo lokalnih optimuma i sedlastih točaka. U tim točkama, gradijent funkcije gubitka je nula, što znači da možemo zapeti u njima. Ta činjenica otežava problem pronalaska minimuma funkcije gubitka te želimo koristiti metodu optimizacije koja neće zaglaviti u sedlastim točkama ili lokalnim minimuma. U nastavku su opisani neki od postupaka koje možemo koristiti u tu svrhu.

Zalet

Metoda zaleta ili momenta (engl. *momentum*) dizajnirana je da ubrza učenje neuronske mreže, pogotovo kad se suočavamo s malim, ali konzistentnim gradijentima ili gradijentima s puno šuma. Učenje s momentom akumulira eksponencijalni pomični prosjek prethodnih gradijenata i nastavlja se kretati u tom smjeru. Akumuliranje prethodnih gradijenata omogućava nam da se nastavimo kretati čak i ako dođemo u točku u kojoj je gradijent jednak nuli.

Uvodimo varijablu $\boldsymbol{v}$ koja akumulira eksponencijalni pomični prosjek prethodnih gradijenata, hiperparametar $\alpha \in [0, 1)$ s kojim množimo prethodno izračunati prosjek. Dobivenim prosjekom u svakom koraku ažuriramo parametre modela kao što je prikazano jednadžbama (2.16) i (2.17):

$$\boldsymbol{v} \leftarrow \alpha \cdot \boldsymbol{v} - \eta \nabla_{\boldsymbol{\theta}} L \tag{2.16}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v} \tag{2.17}$$

Adam

Adam je metoda stohastičke optimizacije s adaptivnim korakom učenja [9]. Temelji se na računanju individualnih adaptivnih koraka učenja za parametre mreže na temelju procjene momenata gradijenta prvog i drugog reda.

Adam akumulira eksponencijalni pomični prosjek gradijenta $\boldsymbol{v}_t$ i kvadrata gradijenta $\boldsymbol{m}_t$, kao što je opisano jednadžbama (2.18) i (2.19):

$$\boldsymbol{v}_t \leftarrow \beta_1 \cdot \boldsymbol{v}_{t-1} + (1 - \beta_1) \cdot \nabla_{\boldsymbol{\theta}} L \tag{2.18}$$

$$\boldsymbol{m}_t \leftarrow \beta_2 \cdot \boldsymbol{m}_{t-1} + (1 - \beta_2) \cdot (\nabla_{\boldsymbol{\theta}} L)^2 \tag{2.19}$$

Hiperparametri $\beta_1, \beta_2 \in [0, 1)$ upravljaju eksponencijalnim pomičnim prosjekom gradijenta i kvadriranog gradijenta. Prosjeci gradijenta i kvadriranog gradijenta zapravo su procjena momenta prvog i drugog reda, no kako se njihove inicijalne vrijednosti inicijaliziraju na 0, procjene su pristrane prema 0. Zbog toga je potrebno ispraviti tu pristranost kao što je prikazano jednadžbama (2.20) i (2.21):

$$\hat{\boldsymbol{v}}_t \leftarrow \frac{\boldsymbol{v}_t}{1 - \beta_1^t} \tag{2.20}$$

$$\hat{\boldsymbol{m}}_t \leftarrow \frac{\boldsymbol{m}_t}{1 - \beta_2^t} \tag{2.21}$$

Promjena parametara u koraku t prikazana je jednadžbom (2.22). Hiperparametar ϵ dodaje se u nazivnik radi numeričke stabilnosti.

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta \cdot \frac{\hat{\boldsymbol{v}}_t}{\sqrt{\hat{\boldsymbol{m}}_t} + \epsilon} \tag{2.22}$$

Uobičajene vrijednosti hiperparametara koje se koriste pri treniranju neuronskih mreža su $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ i $\epsilon = 10^{-8}$.

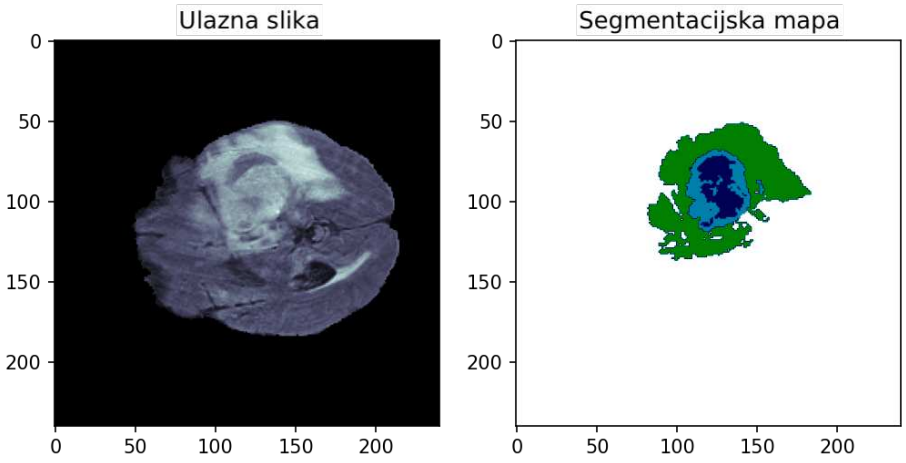
Normalizacija nad grupom

Normalizacija nad grupom (engl. *batch normalization*) metoda je u dubokom učenju koja znatno ubrzava i stabilizira treniranje dubokih modela. Zbog toga što izlaz jednog sloja predstavlja ulaz u sljedeći sloj, čak i mala promjena u izlazima ranih slojeva može imati velik utjecaj na krajnji izlaz. U praksi se pokazalo da neuronske mreže lakše uče na podacima koji su normalizirani, to jest imaju srednju vrijednost 0 i varijancu 1. Normalizacija nad grupom radi tako da normalizira i izlaze svakog sloja tako da sljedeći sloj opet ima ulaze na kojima će lakše učiti. Pritom se normalizacija radi na temelju izlaza koje sloj ima za svaki od uzoraka za učenje iz minigrupe. Dakle, za ulaze $x_1, x_2, ..., x_n$ dobivamo izlaze $o_1, o_2, ..., o_n$. Za dobivene izlaze računamo srednju vrijednost i standardnu devijaciju i na temelju dobivenih vrijednosti izlaze normaliziramo u $\hat{o}_1, \hat{o}_2, ..., \hat{o}_n$ koji postaju novi izlaz sloja. Ako koristimo normalizaciju nad grupom, neuroni ne trebaju imati prag jer će ga normalizacija nad grupom ukloniti.

Mreže koje koriste normalizaciju nad grupom, obično uče u manjem broju iteracija, mogu koristiti veće stope učenja i robusnije su na loše inicijalizirane parametre u odnosu na mreže koje ne koriste normalizaciju nad grupom. Ova poboljšanja objašnjavaju eksperimenti koji sugeriraju da normaliziranje aktivacija dovodi do glađenja pejzaža funkcije cilja [20].

2.3. Semantička segmentacija

Semantička segmentacija jedan je od najpopularnijih problema s kojima se bavi računalni vid. Zadatak je semantičke segmentacije prepoznati i lokalizirati sve što se nalazi na slici. To se postiže dodjeljivanjem oznake svakom pikselu ulazne slike. Semantičku segmentaciju stoga možemo promatrati i kao klasifikaciju na razini piksela. Semantička segmentacija ne razlikuje različita pojavljivanja iste stvari, dakle ako su na slici dva psa, svi pikseli na kojima se nalazi pas bit će odgovarajuće označeni, ali neće biti prepoznato da se radi o dva različita psa. Rezultat je semantičke segmentacije matrica iste rezolucije kao i ulazna slika, gdje svaki element matrice predstavlja oznaku klase za odgovarajući piksel. Takvu matricu nazivamo segmentacijskom mapom. Na slici 2.7 prikazana je ulazna slika i segmentacijska mapa za skup podataka koji sam obrađivao u ovom radu.



Slika 2.7: Primjer ulazne slike i pripadne segmentacijske mape. Ovdje se radi o slici magnetske rezonancije mozga i segmentacijske mape na kojoj su prikazane pojedine strukture tumora prisutnog u mozgu. Bijeli dio segmentacijske mape predstavlja pozadinu, to jest dio slike na kojoj nije prisutan tumor. Zeleno područje označava oticanje oko jezgre tumora. Svjetloplavo područje označava trenutno upaljeni dio jezgre tumora, a tamnoplavo područje označava nekrotični dio jezgre tumora.

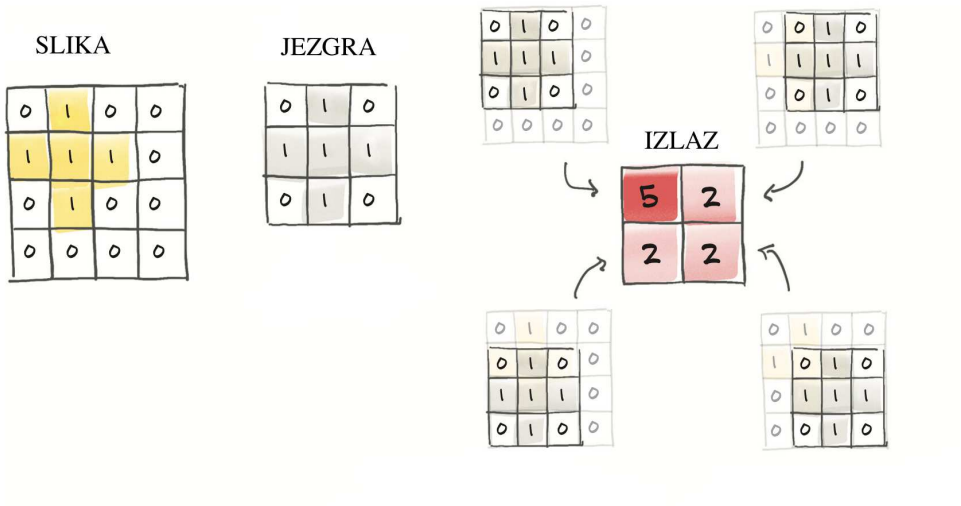
2.4. Konvolucijske neuronske mreže

Za rad sa slikama ili bilo kojom drugom vrstom podataka u kojoj postoje vremenske i prostorne ovisnosti između podataka, potpuno povezani slojevi nisu se pokazali kao učinkovito rješenje. Kada potpuno povezanim slojevima dajemo slike, uzimamo svaki piksel ulaza i računamo njihovu linearnu kombinaciju. S jedne strane tako dopuštamo da kombinacija bilo kojeg piksela bude potencijalno relevantna rješavanju našeg problema. S druge strane, to znači da ne uzimamo u obzir relativni položaj piksela jer cijelu sliku tretiramo kao vektor brojeva. Dakle kada bismo htjeli naučiti prepoznati neki predmet na slici, morali bismo učiti ispočetka svaki put kad bi se predmet nalazio na različitom mjestu u slici. Kažemo da potpuno povezana mreža nije invarijantna na translaciju.

Konvolucijske neuronske mreže pokazale su se učinkovitima u obradi podataka s topologijom rešetke, kao što su 2D i 3D slike. Konvolucijske neuronske mreže su neuronske mreže s barem jednim konvolucijskim slojem.

Diskretna konvolucija

Diskretna konvolucija za slike definirana je kao skalarni produkt matrice težina, koju nazivamo jezgrom konvolucije (engl. *kernel*) sa svakim susjedstvom ulazne slike. Na izlazu konvolucije također dobivamo sliku, to jest mapu značajki.



Slika 2.8: Primjer primjene diskretne konvolucije na 2D slici. Slika je preuzeta iz [21].

Jezgra konvolucije tipično je manja od slike. Elementi izlazne slike, to jest mape značajki ovise o lokalnom susjedstvu elemenata ulazne mape značajki. Svi elementi mape značajki računaju se uz pomoć istog skupa parametara, što znači da dobivamo reprezentaciju ekvivarijantnu s obzirom na pomak. Konvolucija koristi puno manje veza nego potpuno povezani slojevi, što znači da imamo i manje parametara.

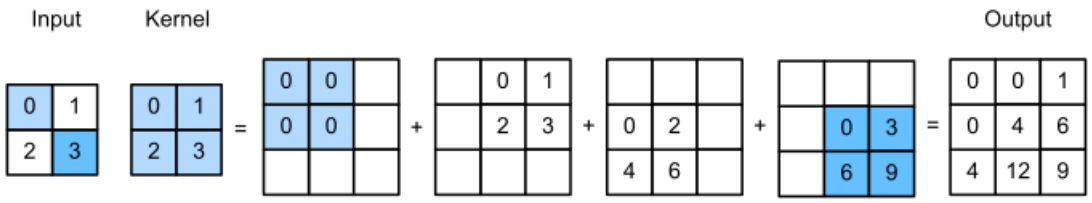
U jednom konvolucijskom sloju obično koristimo više konvolucijskih jezgri. Broj mapa značajki na izlazu odgovara broju korištenih jezgri.

U dubokoj konvolucijskoj mreži, izlazne značajke mogu modelirati interakciju velike regije ulaznih značajki. Skup svih elemenata ulaza koji mogu utjecati na značajku nazivamo receptivnim poljem značajke. Receptivno polje aktivacija konvolucijskih modela raste s dubinom.

Transponirana konvolucija

Transponirana konvolucija ili unatražna konvolucija omogućuje nam da prostorne dimenzije mape značajki, koje smo saželi konvolucijom, vratimo na originalne vrijednosti. To nam je osobito korisno u semantičkoj segmentaciji gdje želimo da je slika na izlazu iste rezolucije kao slika na ulazu. Transponirana konvolucija svaki element

ulazne slike množi sa svakim elementom jezgre i dobivene vrijednosti upisuje u izlazni tenzor te ih zbraja u slučaju preklapanja. Slikovni prikaz operacije transponirane konvolucije prikazan je na slici 2.9.



Slika 2.9: Primjer primjene transponirane konvolucije. Radi se o unatražnom koraku kroz standardnu konvoluciju. Koristi se za uvećanje prostornih dimanzija mapa značajki. Svaki element ulaznog tenzora množimo jezgrom unatražne konvolucije i dobivene vrijednosti smještamo na odgovarajuće mjesto u izlaznoj matrici. Na mjestima preklapanja zbrajamo dobivene vrijednosti. Slika je preuzeta iz [22].

Sažimanje

Ako ulazni podatak želimo prevesti u simboličku kategoriju, nakon konvolucijskog sloja dodajemo sloj sažimanja. Funkcija sažimanja mapira skup prostorno bliskih ulaznih značajki u jednu značajku na izlazu, obično neki statistički pokazatelj sažetog susjedstva, kao što je maksimum ili srednja vrijednost.

Sažimanje čini reprezentaciju invarijantnom na male pomake. Sažimanje također smanjuje dimenzije podataka koji ulaze u sljedeći sloj, što često poboljšava učinkovitost modela i smanjuje količinu memorije potrebnu za spremanje parametara modela.

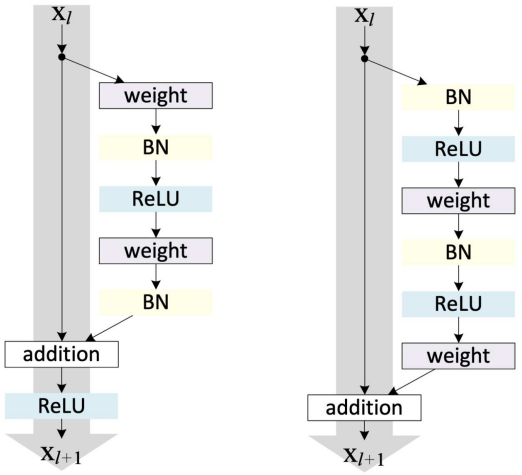
2.4.1. Rezidualne neuronske mreže

U početku razvoja neuronskih mreža, bolji rezultati dobivali su se sve dubljim arhitekturama neuronskih mreža, no pokazalo se da postoji granica dubine nakon koje postaje teže trenirati modele i postiže se manja točnost nego za pliće modele. Kao rješenje tog problema razvijene su rezidualne jedinice neuronske mreže [5]. Ideja je ponovnim korištenjem aktivacija iz prijašnjih slojeva riješiti problem degradacije točnosti dubokih modela.

Rezidualne jedinice karakterizira korištenje preskočnih veza (engl. *skip connections*). One uzimaju ulaz u sloj i zbrajaju ga s izlazom sloja koji se nalazi nekoliko slojeva kasnije. Preskočna veza može biti funkcija identiteta, ali može biti i neko drugo preslikavanje ulaza. Rezidualne jedinice umanjuju problem nestajućih gradijenata koji

se pojavljuje kod dubokih arhitektura te je pokazano da točnost rezidualnih modela raste što je mreža dublja.

U ovom radu, koristio sam dvije vrste rezidualnih jedinica koje su prikazane na slici 2.10.



(a) Originalna rezidualna jedinica

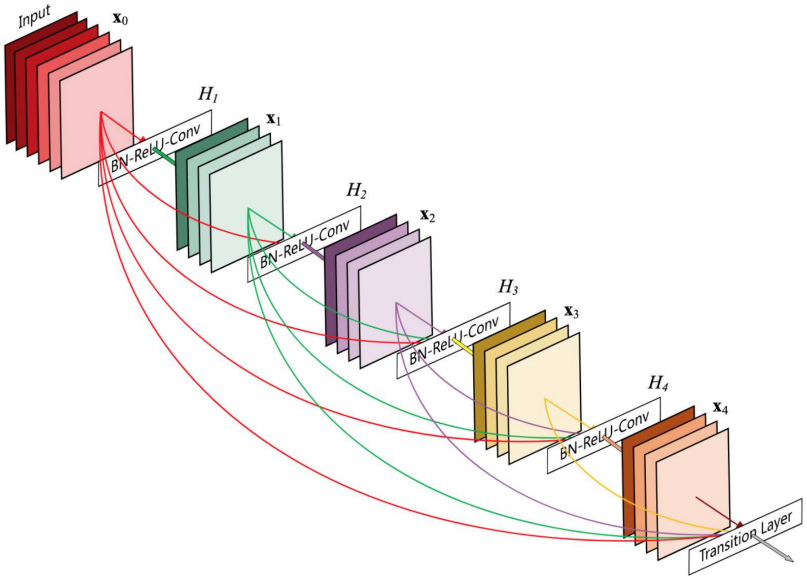
(b) Predaktivacijska rezidualna jedinica

Slika 2.10: Dvije vrste rezidualnih jedinica koje sam koristio u ovom radu. Slike su preuzete iz [6].

2.4.2. Gusto povezane neuronske mreže

Rezidualne neuronske mreže pokazale su da preskočne veze od ranijih prema kasnijim slojevima umanjuju problem nestajućih gradijenata i poboljšavaju performanse dubokih modela. Gusto povezane neuronske mreže (engl. *dense convolutional network*, kratica *DenseNet*) ideju unaprijednih veza vode korak dalje [7]. U blokovima gusto povezane mreže, konvolucijski slojevi na ulaz primaju konkatenirane izlaze svih ranijih slojeva tog bloka i svoje izlaze prosljeđuju svim ostalim slojevima unutar bloka. Na slici 2.11, prikazan je primjer takvog gusto povezanog bloka.

U bloku s L slojeva, gusto povezane neuronske mreže imaju $\frac{L(L+1)}{2}$ veza, za razliku od L veza u tradicionalnim arhitekturama. Pomalo kontraintuitivna posljedica te guste povezanosti je činjenica da gusto povezane mreže obično zahtijevaju manje parametara od tradicionalnih ili rezidualnih dubokih mreža. Dobivene značajke se propagiraju dublje i ponovno se koriste. Tako je maksimiziran tok informacije između slojeva i pojedini slojevi mogu biti uski (obično 32 filtra po konvolucijskom sloju kod modela za



Slika 2.11: Primjer gusto povezanog bloka. Slika je preuzeta iz [7].

velike slike) jer dodaju mali skup značajki kolektivnom znanju mreže koje se prosljeđuje do posljednjeg sloja. Gusto povezane mreže također je lakše trenirati. Uočen je i regularizacijski učinak gustih veza. Gusto povezane neuronske mreže manje su sklone pretrenirati se kada radimo s manjim skupovima podataka u odnosu na tradicionalne arhitekture mreža [7].

3. Korišteni skup podataka

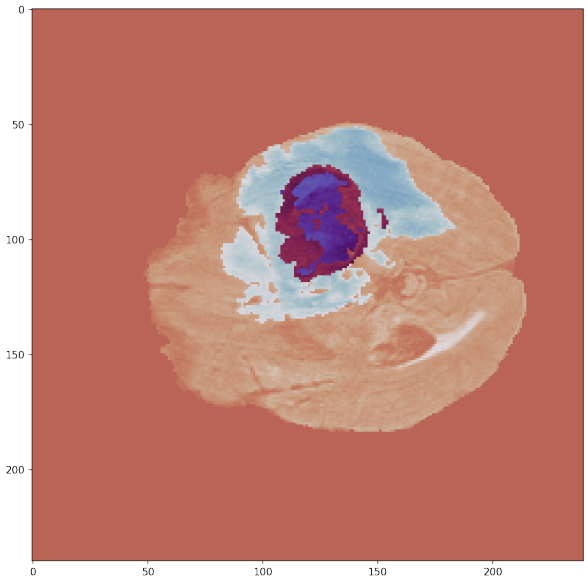
U ovom radu bavit ću se semantičkom segmentacijom novotvorina u slikama magnetske rezonancije. Podatke koje ću koristiti za treniranje i validaciju modela pronašao sam na web stranici `www.kaggle.com`. Radi se o skupu podataka BraTS2020 (kratica za *Brain Tumor Segmentation*). Podaci su skupljeni i pretprocesirani radi korištenja u istoimenom godišnjem natjecanju u semantičkoj segmentaciji tumora i predviđanju stope i vremena preživljavanja pacijenata nakon dijagnoze. Skup podataka detaljnije je opisan u [13], [2] i [3], a u ostatku ovog poglavlja predstavit ću sažetak tih radova.

Tumori prisutni u svim snimkama iz korištenog skupa podataka su gliomi. Radi se o tumorima koji nastaju mutacijama u glija stanicama mozga i kralježničke moždine te se nakon nastanka agresivno šire na okolna tkiva. Gliomi su najčešća, ali i najsmrtonosnija vrsta primarnih tumora mozga, to jest tumora koji počinju u mozgu. U svojim najagresivnijim oblicima, medijan vremena preživljavanja kreće se između 9 i 12 mjeseci [11]. Na mjestima infiltracije tumora u okolno tkivo događa se raspadanje krvno-moždane barijere. Zbog invazivne prirode glioma, gliomi visokog stupnja ne mogu se uspješno odstraniti. S druge strane, gliomi niskog stupnja rastu sporo i mogu se pratiti bez potrebe za liječenjem osim ako uzrokuju simptome pa se zato i odgađa korištenje agresivnih tretmana dok se za time ne pojavi potreba. U korištenom skupu podataka, nalaze se snimke mozgova 293 pacijenata s gliomom visokog stupnja i 76 pacijenata s gliomom niskog stupnja.

U kontekstu ovog skupa podataka, definirane su tri vrste intratumorskih struktura: edem (engl. *peritumoral edema*), invazivna jezgra (engl. *enhancing core*) i nekrotična jezgra (engl. *necrotic core*). Edem je oticanje moždanog tkiva koje se pojavljuje oko tumora. Invazivna jezgra je područje tumora koje trenutno raste. Ona je karakterizirana većom prokrvljenošću zbog raspadanja krvno-moždane barijere. Nekrotična jezgra tumora područje je tumora u kojem su stanice već odumrle i pritom ispustile citoplazmu i njezin sadržaj. Zbog toga se nekrotična jezgra naziva i jezgra ispunjena tekućinom. Za svaki mozak u skupu podataka dostupna je i segmentacijska mapa na kojoj su označene

tri navedene strukture. Na slici 3.1 prikazana je segmentacijska mapa superponirana na krišku snimke mozga. Bijelo područje predstavlja edem, crveno invazivnu jezgru, a ljubičasto nekrotičnu jezgru. Korištene strukture zadovoljavaju određene radiološke kriterije i služe kao identifikatori za automatizirani pronalazak regija, a ne kao biološka interpretacija. To znači da nisu označene samo strukture nastale kao posljedica tumora, već sve strukture te vrste. Na primjer, edem može nastati kao posljedica tumora, ali prisutan je i kao rezultat liječenja.

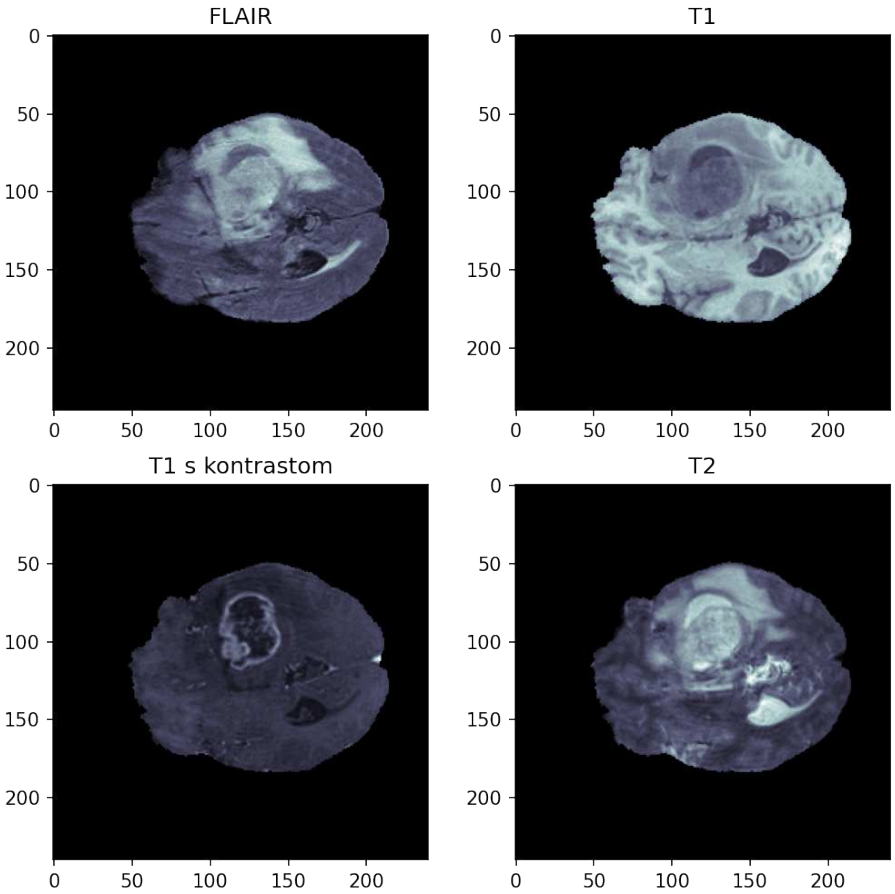
Model koji ću koristiti za segmentaciju tumora na svojem izlazu davat će segmentacijske mape s oznakama koje odgovaraju svakoj od navedenih struktura i pozadini. Za evaluaciju rezultata segmentacije ne koriste se strukture koje su eksplicitno dane u segmentacijskim mapama. Opisane strukture predstavljaju samo vizualne identifikatore, no za kliničke primjene, relevantnije su regije tumora koje uključuju pojedine strukture. Regije koje se koriste za evaluaciju su cijeli tumor, jezgra tumora i invazivna jezgra tumora. Pritom cijeli tumor uključuje sve tri opisane strukture, a jezgra tumora uključuje invazivnu i nekrotičnu jezgru.



Slika 3.1: Segmentacijska mapa superponirana na krišku snimke magnetske rezonancije mozga. Snimka i segmentacijska mapa preuzete su iz skupa podataka BraTS2020. Bijelo područje predstavlja edem, crveno invazivnu jezgru tumora, a ljubičasto nekrotičnu jezgru. Sve ostalo je pozadina, to jest područje bez tumora.

U skupu podataka, za svakog pacijenta dostupne su četiri snimke mozga dobivene

magnetskom rezonancijom u formatu NIfTI (kratica za *Neuroimaging Informatics Technology Initiative*), svaka dobivena različitim načinom snimanja. Načini snimanja korišteni u dobivanju snimki za ovaj skup podataka su snimanje mjerenjem vremena T1-relaksacije, snimanje mjerenjem vremena T2-relaksacije, snimanje mjerenjem T1-relaksacije s ubrizganim kontrastom na bazi gadolinija te FLAIR (kratica za *fluid attenuated inversion recovery*). Na slici 3.2 prikazane su snimke dobivene različitim načinima snimanja.



Slika 3.2: Ista kriška mozga u svakom od korištenih načina snimanja. Na T2 i FLAIR snimkama najjasnije se vidi edem. Invazivni i nekrotični dijelovi jezgre tumora najistaknutiji su na T1 snimci s kontrastom. Pripadna segmentacijska maska koja označava gdje se nalazi tumor prikazana je na slici 3.1.

Različiti načini snimanja su korišteni jer se na njima jasnije vide različite strukture. Edem je najvidljiviji na T2 i FLAIR snimkama. Invazivna jezgra tumora najvidljivija

je na T1 snimkama s kontrastom budući da je invazivno područje najbolje prokrvljeno, a kontrast se nakuplja u krvi. Zbog jarkog ruba koji označava invazivnu jezgru, na T1 snimkama s kontrastom također se jako dobro vidi i nekrotična jezgra kao područje niskog intenziteta unutar tog ruba.

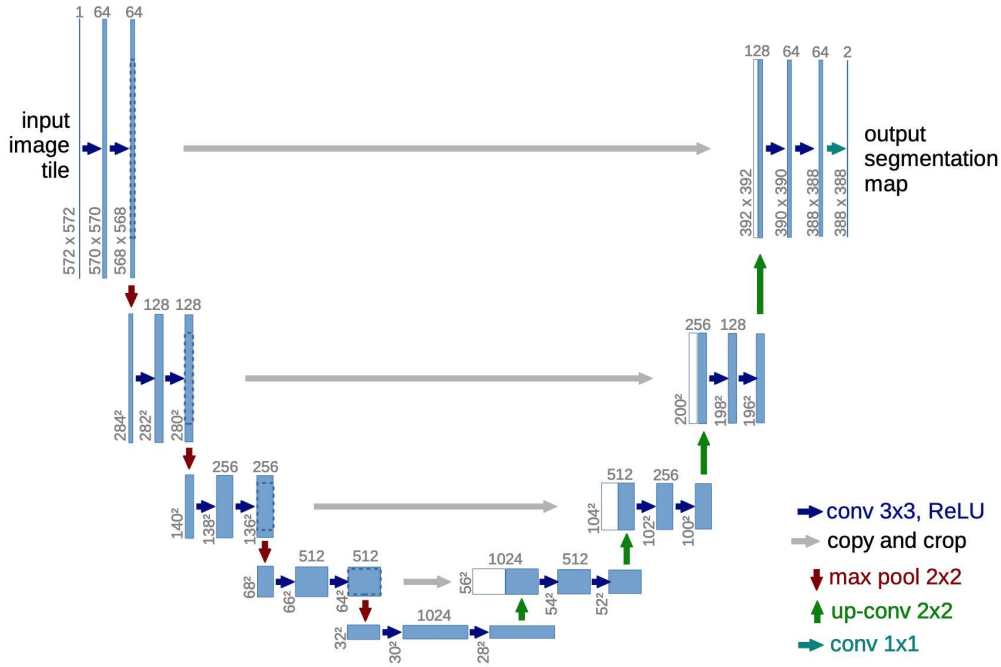
Različiti načini snimanja mozga moraju se provoditi zasebno, što znači da pacijent nije nužno bio u potpuno istom položaju prilikom svakog snimanja. Da bi ujednačili podatke, autori skupa podataka sve su načine snimanja za jednog pacijenta registrirali koristeći T1 snimku s kontrastom kao referentnu snimku. Također, sa svih snimki je uklonjena lubanja jer nije relevantna za segmentaciju tumora. Snimke različitih pacijenata nisu međusobno registrirane. Da bi ispravili nejednakosti u rezolucijama snimki do kojih je došlo jer su pacijenti snimani u različitim institucijama s različitim uređajima, sve snimke su ponovno uzorkovane na izotropnu prostornu rezoluciju od 1 mm^3. Svaka snimka ima 155 krišaka rezolucije 240x240.

Metapodaci o dobi pacijenata, vremenu preživljavanja i stupnju tumora dostupni su u csv datotekama. U okviru ovog rada, koristio sam samo podatke o stupnju tumora svakog od pacijenata.

4. Korištene arhitekture mreža

4.1. U-Net

U-Net je arhitektura neuronske mreže koja se često koristi za segmentaciju biomedicinskih slika, čime se i ovaj rad bavi [18]. Obično kada radimo s biomedicinskim slikama, problem predstavlja mala veličina ulaznog skupa podataka zbog ograničene dostupnosti slika. U-Net je modifikacija potpuno konvolucijskog modela koja dobro generalizira i kad ima mali broj slika za treniranje na raspolaganju.



Slika 4.1: Primjer U-neta. Slika je preuzeta iz [18].

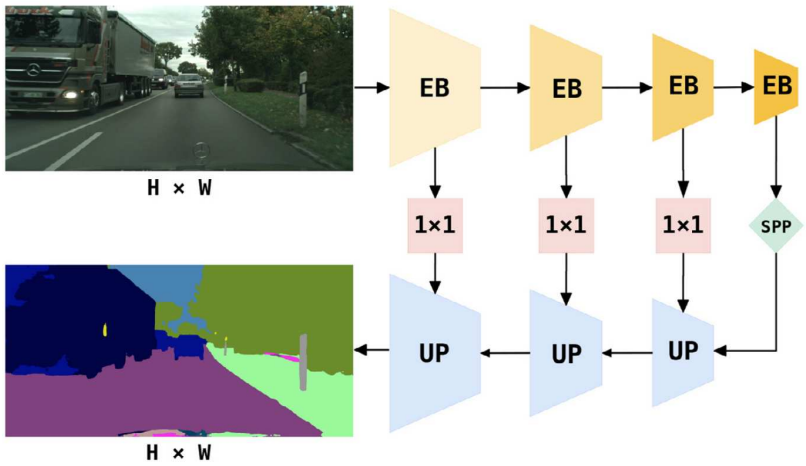
Bitna značajka U-neta dvije su podatkovne putanje koje čine ovaj model. Na lijevoj strani slike 4.1 je koder, koji osigurava točna predviđanja, a s desne strane je dekoder koji vraća informacije o detaljima slike. U-Net također karakterizira velik broj mapa značajki u blokovima za naduzorkovanje, što omogućuje tok informacije o kontekstu

prema slojevima više rezolucije. Zbog velikog broja mapa značajki u slojevima dekodera, dekoder i koder su simetrični, zbog čega prikaz arhitekture mreže podsjeća na veliko slovo U, po čemu je ova arhitektura i dobila ime.

Svaki blok kodera sastoji se od dvije 3x3 konvolucije nakon koje slijedi sloj sažimanja koristeći maksimum (engl. *max pooling*) 2x2 s korakom 2 radi smanjenja rezolucije. Svaki blok dekodera sastoji se od naduzorkovanja mapa značajki pomoću 2x2 transponirane konvolucije, konkatenacije s odgovarajućim izlazom bloka kodera i dvije 3x3 konvolucije. Zadnji korak mreže je 1x1 konvolucija koja preslikava posljednje mape značajke u željeni broj kanala izlaza.

4.2. SwiftNet

SwiftNet je model s ljestvičastim naduzorkovanjem za brzu i učinkovitu semantičku segmentaciju slika [15]. Arhitektura mreže oblikovana je tako da kao kralježnicu kodera koristi model koji je predtreniran na ImageNet skupu podataka. Predtrenirani početni parametri služe kao još jedna metoda regularizacije. Tri osnovna dijela SwiftNeta su koder, dekoder i modul za povećanje receptivnog polja.

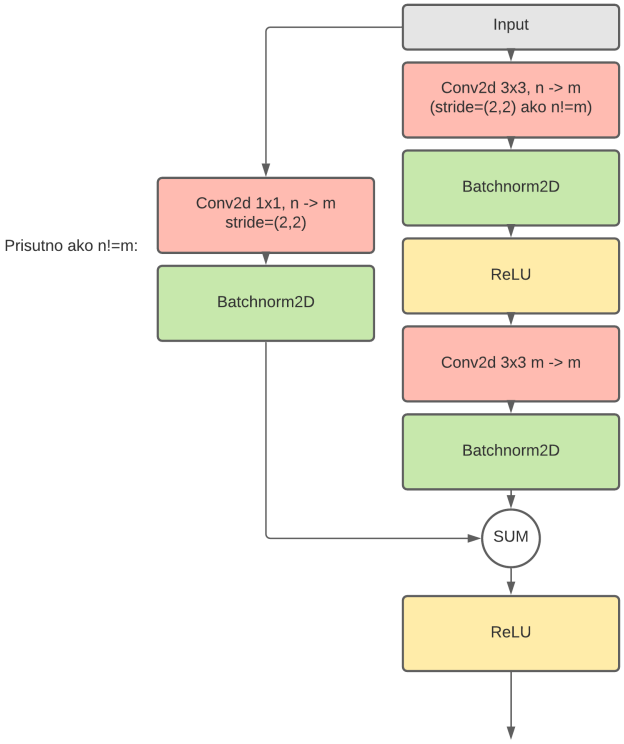


Slika 4.2: Prikaz arhitekture SwiftNeta. Slika je preuzeta iz [16].

4.2.1. ResNet18

Implementacija SwiftNeta koju sam koristio u ovom radu kao koder koristi model ResNet18, što je rezidualni model s 18 konvolucijskih slojeva. Prvi korak te mreže je konvolucija s jezgrom veličine 7x7 i korakom 2 koja 4 ulazna kanala slike preslikava u 64

mape značajki. Nakon toga primjenjuju se normalizacija nad grupom, zglobnica i 2x2 sažimanje maksimalnom vrijednošću. Zatim slijede 4 rezidualna bloka koji su na slici 4.2 predstavljeni narančastim trapezima. Rezidualni blokovi sastoje se od dviju rezidualnih jedinica koje sadržavaju svaka po dvije 3x3 konvolucije. U svakom bloku osim prvog, prva rezidualna jedinica udvostručuje broj mapa značajki i smanjuje prostornu rezoluciju za faktor 2. Izlaz svakog rezidualnog bloka prosljeđuje se odgovarajućem bloku dekodera lateralnom vezom. Pritom je bitno napomenuti da se bolji rezultat ostvaruje ako lateralnom vezom proslijedimo izlaz sume prije nego što ga propustimo kroz zglobnicu.



Slika 4.3: Korištena rezidualna jedinica, u velikoj mjeri odgovara rezidualnoj jedinici prikazanoj na slici 2.10a. Rezidualni blokovi koji su na slici 4.2 prikazani narančastim trapezima sastoje se od dviju ulančanih rezidualnih jedinica. Ulazni broj mapa značajki označen je s n, a izlazni s m. Svaki rezidualni blok osim prvog dvostruko povećava broj mapa značajki. Prva rezidualna jedinica svakog bloka zadužena je za povećavanje broja mapa značajki. Zbog toga je u prvoj rezidualnoj jedinici broj m dvostruko veći od broja n te je umjesto jednostavne preskočne veze, kao što je prikazano na slici 2.10a, potrebno primijeniti 1x1 konvoluciju za povećanje broja mapa značajki i smanjivanje prostorne rezolucije.

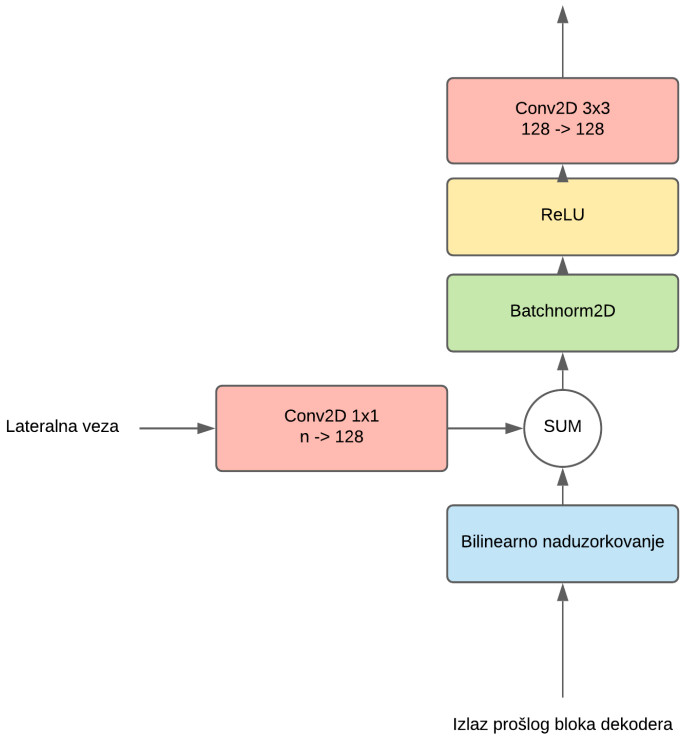
4.2.2. Dekoder

Dekoder transformira semantički bogatu sliku niske rezolucije koje dobivamo na izlazu kodera u sliku rezolucije jednake ulaznoj. Korišteni blokovi dekodera na slici 4.2 prikazani su plavim trapezima. Svaki blok dekodera bilinearno naduzorkuje mape značajki koje dobiva na ulazu. Osim mape značajki prošlog sloja, svaki blok dekodera lateralnom vezom dobiva mape značajki iz odgovarajućeg rezidualnog sloja koje propuštamo kroz 1x1 konvoluciju. Tom konvolucijom podešavamo broj mapa značajki tako da bude jednak broju mapa značajki na ulazu u blok dekodera. Dobivene lateralne mape značajki zbrajamo s bilinearno naduzorkovanim mapama značajki prethodnog sloja naduzorkovanja i nad dobivenim tenzorom primjenjujemo 3x3 konvoluciju koja izvlači značajke iz svih dosad skupljenih informacija na trenutnoj rezoluciji.

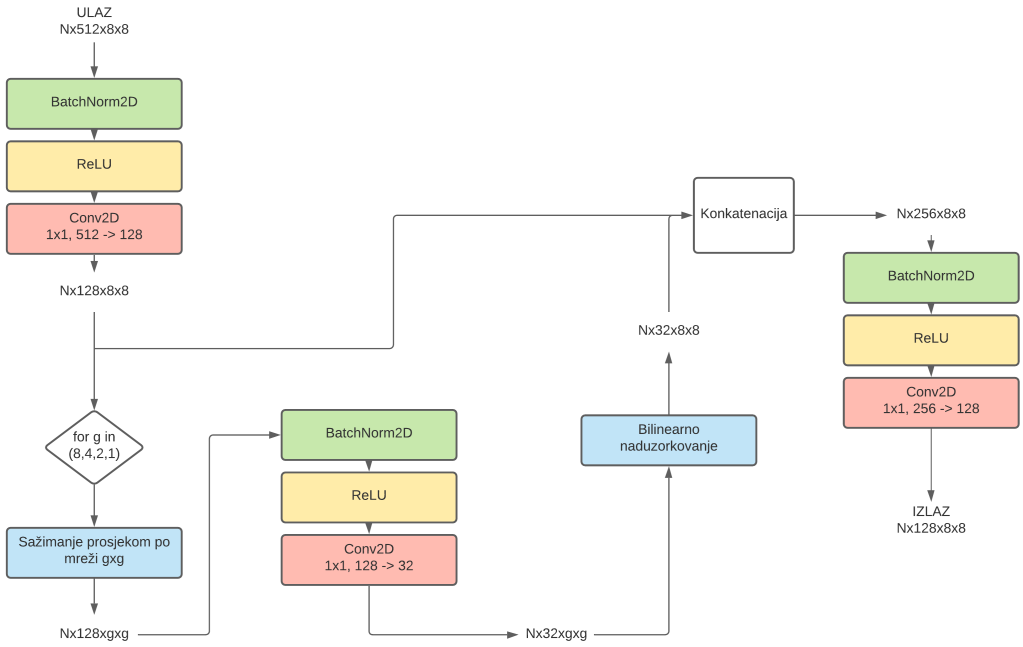
Za razliku od U-Neta, dekoder i koder u SwiftNetu su asimetrični. Koder primjenjuje više konvolucija po sloju, dok dekoder primjenjuje samo jednu. Osim toga, broj mapa značajki kodera raste tijekom putanje podataka, dok je broj mapa značajki dekodera jednak u svakom bloku.

4.2.3. Modul za povećanje receptivnog polja

Između dekodera i kodera nalazi se modul koji povećava receptivno polje. U implementaciji SwiftNeta koju sam koristio u ovom radu, to je SPP modul (kratica za *spatial pyramid pooling*). Korištena je implementacija SPP modula predstavljena u [10]. Ulaz u SPP modul najprije se projicira 1x1 konvolucijom na 4 puta manje mapa značajki. Dobiveni izlaz zatim se projicira na 4 puta manji broj mapa značajki i sažima prosjekom na rešetke dimenzija 1x1, 2x2, 4x4 i 8x8 koje se bilinearno naduzorkuju na rezoluciju ulaza. Dobivene rešetke zatim se konkateniraju s projiciranim ulazom i na dobiveni tenzor primjenjuje se 1x1 konvolucija koja na izlazu daje 128 mapa značajki.



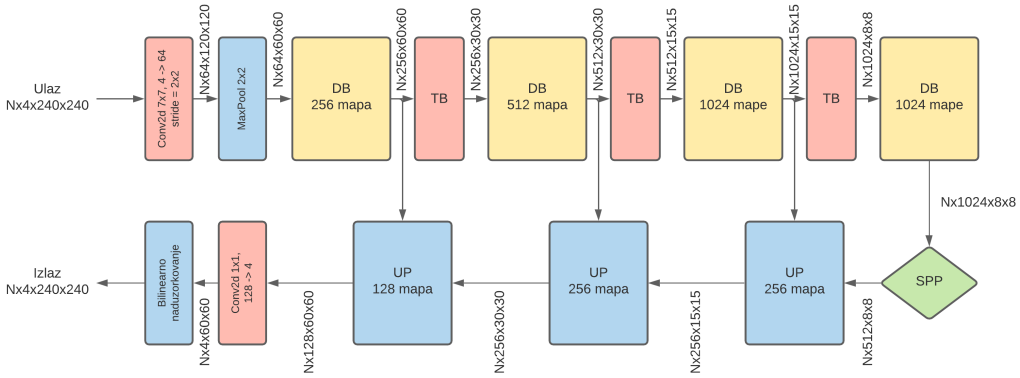
Slika 4.4: Korišteni blok dekodera. Na slici 4.2 prikazan je plavim trapezom. Blokovi dekodera zaduženi su za naduzorkovanje i izvlačenje značajki iz svih dosad skupljenih informacija na trenutnoj rezoluciji. Varijabla n na slici predstavlja broj mapa značajki odgovarajućeg rezidualnog bloka kodera. Kako je broj mapa značajki u svakom bloku dekodera jednak 128, izlaze odgovarajućeg bloka kodera koje dobivamo lateralnom vezom moramo propustiti kroz 1x1 konvoluciju koja namješta broj mapa značajki na 128. Te 1x1 konvolucije na slici 4.2 prikazane su ružičastim kvadratima.



Slika 4.5: Korišteni SPP modul. Na slici 4.2 prikazan je zelenim rombom. SPP modul koristio sam da bih povećao receptivno polje slojeva koji slijede. Na slici N predstavlja veličinu grupe.

4.3. Gusto povezani model s ljestvičastim naduzorkovanjem

Posljednji model koji sam koristio za semantičku segmentaciju novotvorina gusto je povezani model s ljestvičastim naduzorkovanjem (engl. *DenseNet with ladder-style upsampling*) [10]. Kao i korišteni SwiftNet, ovaj model sastoji se od dvije podatkovne putanje, to jest kodera i dekodera između kojih se nalazi SPP modul radi povećanja receptivnog polja.



Slika 4.6: Arhitektura korištenog gusto povezanog modela s ljestvičastim naduzorkovanjem. Slojevi označeni s DB su gusto povezani slojevi, slojevi označeni s TB su tranzicijski blokovi, a slojevi označeni s UP su blokovi dekodera slični opisanima na slici 4.4. Blokovi dekodera gusto povezanih modela razlikuju se samo u tome što zbrajamo izlaz s ulazom kojeg smo projicirali na odgovarajući broj značajki. Tako efektivno dobivamo rezidualni blok. SPP modul jednak je opisanom na slici 4.5.

4.3.1. DenseNet121

U ovoj arhitekturi kao koder korišten je DenseNet121, to jest gusto povezana neuronska mreža sa 121 konvolucijskim slojem. Kao i ResNet18, DenseNet121 počinje 7x7 konvolucijom s korakom 2 koji na izlazu daje 64 mape značajki i zatim sažimanjem maksimalnom vrijednošću. Nakon toga slijede 4 gusto povezana bloka između kojih se nalaze tranzicijski blokovi. Svaki gusto povezani blok sastoji se od konkatenacije niza gusto povezanih jedinica. Svaka gusto povezana jedinica sastoji se od jedne 1x1 konvolucije i jedne 3x3 konvolucije prije kojih se primjenjuje normalizacija nad grupom i zglobnica. Svaka jedinica na ulaz prima konkatenaciju ulaza u konvolucijski blok i izlaza svih prethodnih jedinica u istom bloku. Tranzicijski blokovi između gusto

povezanih blokova sastoje se od 1x1 konvolucije koje dvostruko smanjuju broj mapa značajki nakon kojeg slijedi 2x2 sažimanje prosjekom.

4.3.2. Dekoder

Zadatak dekodera i ovdje je vratiti detalje koji su se izgubili poduzorkovanjem u koderu. Blokovi dekodera u ovoj arhitekturi zapravo su rezidualni blokovi. Kao i blokovi dekodera u SwiftNetu, na ulaz svakog bloka dolazi reprezentacija niske rezolucije koja je izlaz prethodnog bloka dekodera, ali i izlaz odgovarajućeg gusto povezanog bloka koji dolazi lateralnom vezom. Reprezentaciju niske rezolucije bilinearno naduzorkujemo za faktor 2, a mape značajki koje dolaze lateralnom vezom projiciramo 1x1 konvolucijom na odgovarajući broj značajki. Te dvije reprezentacije zatim sumiramo. Na dobivenu sumu po potrebi primjenjujemo 1x1 konvoluciju za smanjenje broja mapa značajki. Nakon toga primjenjujemo 3x3 konvoluciju i zatim zbrajamo s ulazom bloka dekodera kojeg smo projicirali na odgovarajući broj značajki.

4.3.3. Modul za povećanje receptivnog polja

Nakon prolaska kroz DenseNet121, podaci ulaze u SPP modul, koji je gotovo identičan opisanome u prethodnom potpoglavlju. Jedina razlika je u prvom konvolucijskom sloju koji projicira dobivene značajke na 2 puta manji broj mapa značajki, umjesto na 4 puta manji broj.

4.3.4. Smanjivanje memorijskog zauzeća

Za modele s puno parametara, kao što je ovaj, memorijsko zauzeće predstavlja problem. Kako na raspolaganju obično imamo ograničenu količinu memorije grafičkog procesora, ako radimo s prevelikim modelom, često nećemo moći koristiti velike grupe podataka. To bi moglo rezultirati manje preciznim procjenama gradijenta i nestabilnim statistikama normalizacije nad grupom. Smanjivanje memorijskog zauzeća prilikom treniranja modela možemo postići podešavanjem mreže tako da ne sprema sve aktivacije prilikom unaprijednog prolaska. Umjesto toga, aktivacije se ponovno računaju tijekom unatražnog prolaza. Ova metoda naziva se *gradient checkpointing*. Tijekom unaprijednog prolaza sprema se samo određeni skup aktivacija na temelju kojeg ćemo moći izračunati ostale. U ovom modelu, spremaju se samo izlazi iz 3x3 konvolucija u koderu jer se radi o operacijama koje je najteže izračunati. Prilikom unatražnog prolaza, aktivacije slojeva između 3x3 konvolucija ponovno se računaju i spremaju u

lokalni međuspremnik. Nakon ponovnog računanja aktivacija, provodi se standardni unatražni prolaz. Nakon što obradimo segment mreže između dvije susjedne 3x3 konvolucije čije izlaze smo spremili, lokalni međuspremnik se oslobađa. Sve aktivacije u dekoderu sam spremao, to jest u dekoderu nisam koristio *gradient checkpointing*.

5. Programska implementacija

5.1. Korištene tehnologije

Izvedba praktičnog dijela ovog rada napravljena je u programskom jeziku Python. Radni okvir za automatsku diferencijaciju korišten u ovom radu je PyTorch. Neke od funkcionalnosti PyTorcha koje pogoduju izradi ovog rada su lako rukovanje višedimenzionalnim poljima, to jest tenzorima, ubrzanje izvođenja koda koristeći grafički procesor i jednostavno modeliranje neuronskih mreža. Osim PyTorcha, za manipulaciju višedimenzionalnih polja korištena je biblioteka NumPy. Biblioteku torchvision koristio sam za transformacije korištene pri uvećanju podataka te za učitavanje predtreniranih parametara modela. Za učitavanje snimki mozgova u NIfTI formatu, korištena je biblioteka NiBabel, a za učitavanje metapodataka o pacijentima korištena je biblioteka pandas. Za vizualizaciju rezultata korištena je biblioteka Matplotlib.

Budući da nemam lokalni pristup sklopovlju na kojem bi se modeli korišteni u ovom radu utrenirali u razumnom vremenu, koristio sam uslugu za izvođenje koda u oblaku koju nudi stranica Kaggle. Interakciju s Kaggleovom okolinom provodim preko takozvanih Kaggleovih jezgri (engl. *Kaggle kernel*) koje objedinjuju Jupyterove bilježnice, pristup javno dostupnim skupovima podataka i pristup besplatnim računalnim resursima, uključujući i grafički procesor. Korisnicima je dostupno 73 GB memorije na disku, 13 GB radne memorije i pristup NVIDIA Tesla P100 grafičkom procesoru sa 16 GB memorije. Pritom, svaki korisnik može koristiti grafički procesor 30 sati tjedno, s uvjetom da jedna neprekinuta sesija korištenja smije trajati maksimalno 9 sati.

5.2. Priprema i pristup podacima

Za upravljanje podacima koristio sam klase *Dataset* i *DataLoader* koje nudi radni okvir PyTorch. Prije svega, odredio sam koji pacijenti pripadaju u skup za treniranje, validaciju i testiranje koristeći dostupne metapodatke kako bi omjer pacijenata s tu-

morom visokog i niskog stupnja bio jednak u sva tri podskupa. Skup za treniranje sadrži 263 snimke mozga, a skupovi za validaciju i testiranje svaki po 53 snimke. Radi pristupa podacima, napravio sam potklasu klase *Dataset* koja prima popis puteva do snimki. Na ulaze modela dovodim snimku mozga s 4 kanala, jedan za svaki način snimanja mozga koji je dostupan u skupu podataka.

Za 2D modele, trebamo pristupiti križkama snimki. Pojedinoj križi ne možemo pristupiti bez učitavanja cijele snimke, što predstavlja problem jer bi tako prilikom oblikovanja grupa potencijalno trebali učitati mnogo snimki, a veliki dio svake snimke ostao bi neiskorišten. Također, kako je svaki skup 4 slike i segmentacijske mape za jednog pacijenta veličine 77 MB, u dostupnu radnu memoriju ne stane puno snimki. Zbog toga učitavam slike tako da učitam jednu cijelu snimku i ako to nije dovoljno za ispuniti traženu veličinu grupe, učitavam druge snimke dok ne dosegnem željeni broj križaka. Ako u jednom učitavanju preostane neiskorištenih križaka, spremam ih u podatkovni član razreda *Dataset* i iskorištavam prilikom sljedećeg učitavanja grupe križaka. Kada pristupam snimkama u skupu za treniranje i validaciju, izdvajam samo križe koje sadrže bar jedan piksel tumora, a za testiranje koristim cijele snimke. Snimke normaliziram tako da je srednja vrijednost piksela 0, a varijanca 1 koristeći procjene srednje vrijednosti i varijance na temelju pojedinog pacijenta. Za uvećanje podataka u skupu za testiranje koristim slučajno horizontalno zrcaljenje s vjerojatnošću 0.5.

5.3. 2D modeli

Za segmentaciju glioma u snimkama mozga na temelju pojedinih križi snimke, koristio sam U-Net, SwiftNet i DenseNet121 s ljestvičastim naduzorkovanjem koje sam opisao u prethodnom poglavlju. Jedino odstupanje od danog opisa je koje sam morao napraviti je dvostruko smanjiti broj mapa značajki u svakom konvolucijskom sloju U-Neta jer je arhitektura prikazana slikom 4.1 zauzimala previše memorije. Moje implementacije SwiftNeta i DenseNeta s ljestvičastim naduzorkovanjem temelje se na implementacijama dostupnima u github repozitorijima koda za radove u kojima su arhitekture predstavljene[1][2].

Kao gubitak, koristio sam unakrsnu entropiju s relativnim težinama 1 za pozadinu, 2 za oticanje, 3 za nekrotičnu jezgru i 5 za invazivnu jezgru. Koristio sam optimizator Adam s podrazumijevanim parametrima i L^2 regularizacijom s relativnim doprinosom 0.0005. Za smanjivanje koraka učenja koristio sam kosinusno kaljenje s minimalnom

[1] `https://github.com/orsic/swiftnet`

[2] `https://github.com/ivankreso/ladder-densenet`

vrijednošću koraka učenja 10^{-6}. Početna vrijednost koraka učenja je 10^{-4} za U-Net i SwiftNet, a $5 \cdot 10^{-4}$ za DenseNet.

Kako SwiftNet i DenseNet121 s ljestvičastim naduzorkovanjem kao koder koriste arhitekture koje se često koriste za semantičku segmentaciju, njihove parametre inicijalizirao sam vrijednostima predtreniranima na ImageNet skupu podataka. Pritom, moj se prvi konvolucijski sloj razlikuje od standardne implementacije jer moja ulazna slika ima 4 kanala, a predtrenirane mreže rade na RGB slikama koje imaju 3 kanala. Zbog toga svaki kanal jezgre svojeg prvog konvolucijskog sloja inicijaliziram prosjekom jezgre predtreniranog modela po kanalima. Za sve inicijalizirane slojeve koristio sam dvostruko manji korak učenja nego za sve ostale slojeve.

5.4. 3D modeli

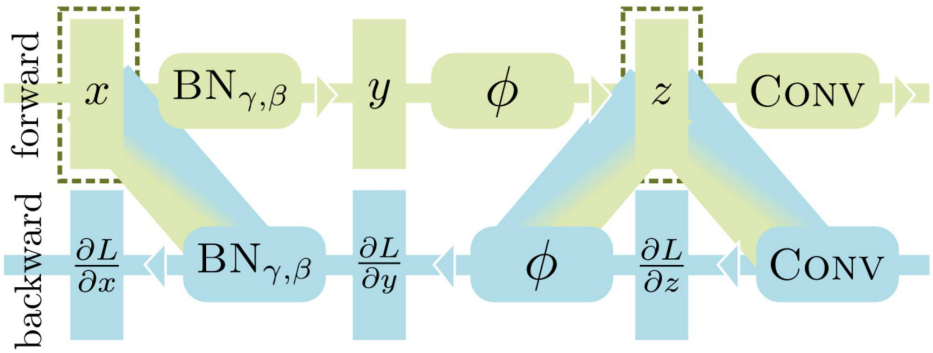
Pokušao sam utrenirati i model za segmentaciju glioma koji će na ulazu prihvaćati čitavu 3D snimku umjesto 2D krišaka mozga. Koristio sam 3D adaptaciju ResNeta opisanog u prethodnom poglavlju. Dakle, svi slojevi ostali su isti, samo rade s 3D slikama umjesto 2D slika. Osim toga, pokušao sam utrenirati i ResNet s dodanim SPP modulom. Način treniranja ovog modela isti je kao i za 2D modele, jedino sam morao napraviti neke promjene u samoj arhitekturi modela kako bih smanjio njihovo memorijsko zauzeće. Također, mreže koje su predtrenirane na ImageNet skupu podataka obično su mreže s 2D slojevima. Zbog toga za 3D modele nisam koristio inicijalizaciju parametrima predtreniranima na ImageNet skupu podataka.

5.4.1. Smanjivanje memorijskog zauzeća modela

Kako se radi o 3D modelima, parametri modela zauzimaju puno više memorije. Također, same snimke zauzimaju puno memorije pa možemo raditi puno manje grupe, što znači da će i procijenjeni gradijenti biti manje precizni. Da bih mogao koristiti veće grupe podataka pri treniranju, napravio sam neke modifikacije na predstavljenom modelu.

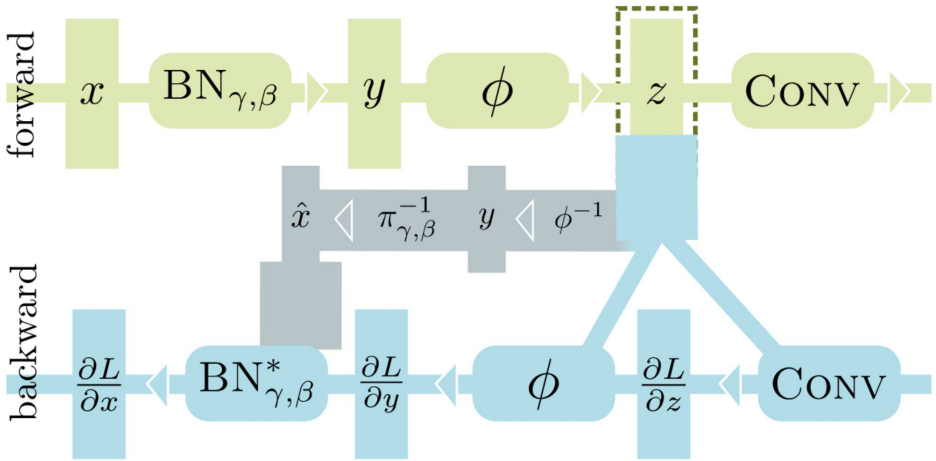
Prvo, prepolovio sam broj mapa značajki u svim konvolucijskim slojevima modela. To je smanjilo memorijsko zauzeće, no ne dovoljno.

U trenutnoj implementaciji normalizacije nad grupom, potrebno je čuvati dva velika međuspremnika, ulaze u normalizaciju nad grupom i ulaze u konvoluciju jer implementacija unatražnog prolaza te ulaze koristi za računanje gradijenata. Naravno, spremanje tih ulaza zauzima puno memorije i željeli bismo nekako smanjiti to zauzeće me-



Slika 5.1: Standardna gradivna jedinica mreže s normalizacijom po grupi. Slika je preuzeta iz [19].

morije.Integrirana normalizacija po grupi i nelinearna aktivacija (engl. *in-place activated batch normalization*, kratica ABN) omogućuje smanjenje memorijskog zauzeća optimizacijom normalizacije nad grupom [19]. ABN smanjuje potrebnu količinu memorije tako što unatražni prolaz nad normalizacijom nad grupom definira preko izlaza normalizacije nad grupom pa se ne moraju pamtiti ulazi u normalizaciju nad grupom, na slici 5.1 prikazani s x, već je dovoljno pamtiti samo ulaze u konvoluciju, na slici 5.1 prikazani sa z.



Slika 5.2: Gradivna jedinica mreže s ABN-om. Slika je preuzeta iz [19].

Umjesto normalizacije nad grupom, u 3D adaptaciji ResNeta koristio sam ABN. Budući da prijenosna funkcija treba biti invertibilna da bismo mogli izračunati vrijednost x u unatražnom prolazu, ne možemo koristiti zglobnicu jer je ona invertibilna samo na intervalu $[0, \infty)$. Zbog toga koristimo propusnu zglobnicu s hiperparametrom $\alpha = 0.01$. Također, umjesto standardnih rezidualnih jedinica morao sam koristiti pre-

daktivacijske rezidualne jedinice u 3D adaptaciji ResNeta jer je ABN dizajniran za rad s predaktivacijskim ResNetom.

6. Eksperimenti

6.1. Mjera točnosti

Kao što je predloženo u [13], za evaluaciju modela koristio sam Diceov koeficijent sličnosti (engl. *Dice score*). Diceov koeficijent sličnosti definiran je za dva skupa, X i Y te se računa prema jednadžbi (6.1):

$$DSC = \frac{2 \cdot |X \cap Y|}{|X| + |Y|} \tag{6.1}$$

Budući da model evaluiramo za tri regije, za svaku regiju zasebno računamo Diceov koeficijent. U kontekstu ovog rada, dva skupa su skup piksela za koje je model odredio da pripadaju određenoj regiji te skup piksela koji zaista pripadaju određenoj regiji prema oznakama iz podataka. Kardinalitet presjeka u ovom slučaju označava ispravno klasificirane piksele, a kardinalitet pojedinog skupa označava broj piksela kojima je pridružena oznaka regije za koju računamo Diceov koeficijent.

6.2. 2D modeli

Modeli čiji su rezultati predstavljeni u ovom odjeljku predviđali su segmentacijske mape za svaku krišku volumena, neovisno o drugim kriškama. Takvim pristupom gubimo prostorni odnos između različitih krišaka i receptivno polje u trećoj dimenziji koje bi moglo dati dodatne informacije koji bi poboljšali rezultate. Unatoč tome, 2D modeli daju dobre rezultate i imaju dodatnu prednost što su manji i lakše se treniraju.

6.2.1. Validiranje ImageNet inicijalizacije na SwiftNet-RN18

Iako su svi modeli korišteni za segmentaciju tumora, inicijalizacija parametara predtreniranima na ImageNet skupu podataka poboljšala je performanse modela. Inicijalizaciju sam validirao u dva koraka, najprije sam inicijalizirao sve slojeve kralježnice modela osim početne konvolucije, a u drugom koraku sam osim toga inicijalizirao

početnu konvoluciju prosjekom predtreniranih jezgri po kanalu. Rezultati su dani u tablici 6.1.

Tablica 6.1: Rezultati validiranja ImageNet inicijalizacije na SwiftNet-RN18. U drugom retku prikazani su rezultati modela u kojemu su svi slojevi kralježnice osim prve konvolucije inicijalizirani parametrima predtreniranima na ImageNet skupu podataka. U trećem retku prikazani su rezultati modela u kojemu su svi slojevi inicijalizirani parametrima predtreniranima na ImageNet skupu podataka. Pritom je prvi konvolucijski sloj inicijaliziran prosjekom predtreniranih jezgri po kanalu.

Varijanta	Srednji Diceov koeficijent		
	Cijeli tumor	Jezgra tumora	Invazivna jezgra
Bez inicijalizacije	0.8782	0.7305	0.7584
Osnovna inicijalizacija	0.9048	0.8177	0.7818
Inicijalizacija prosjekom	0.9036	0.8277	0.7970

Inicijalizacija predtreniranim parametrima uvelike povećava točnost našeg modela na validacijskom skupu, iako su parametri predtrenirani na potpuno nevezanom skupu slika. Čak je i inicijalizacija početne konvolucije prosjekom jezgri povećala točnost, iako ne toliko dramatično kao inicijalizacija ostatka kodera.

6.2.2. Prikaz dobivenih rezultata i usporedba s drugim radovima

U tablici 6.2 navedene su srednje vrijednosti Diceovih koeficijenata dobivene na skupu za testiranje. Točnost je puno manja nego na podskupu za validaciju djelomično zbog pretreniranosti hiperparametara na skup za validaciju, ali vjerojatno i zbog toga što sam model validirao samo na kriškama za koje znamo da sadrže tumor, a testirao sam ga na svim kriškama pojedinog mozga.

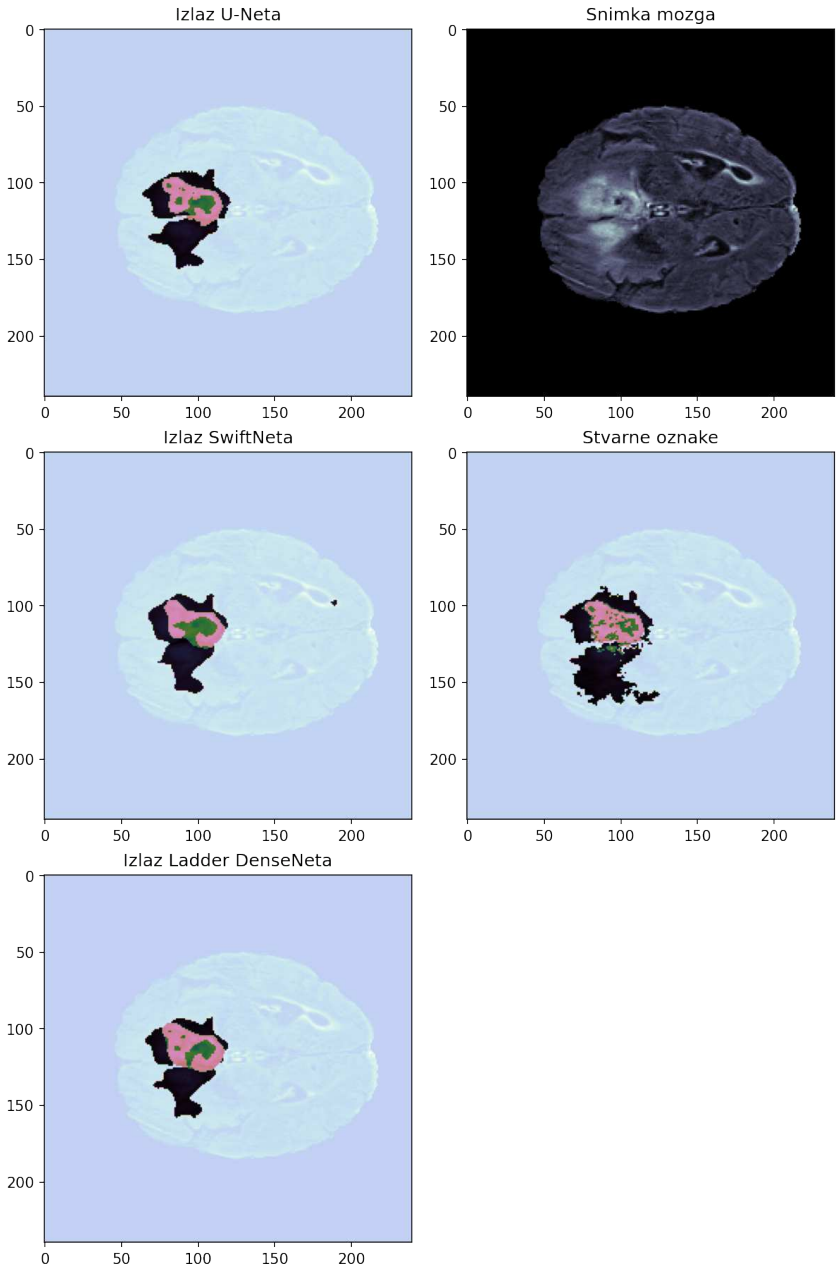
Radi usporedbe, u donja dva retka tablice stavio sam rezultate koje su postigli drugi radovi koji su se bavili mojim skupom podataka i učili su na temelju nezavisnih 2D krišaka. Prilikom usporedbe treba uzeti u obzir da su navedeni radovi koristili službene podatke za testiranje kojima ja nisam imao pristup jer su oni dostupni samo tijekom trajanja natjecanja BraTS. Zato nisam mogao svoje modele testirati na potpuno istim podacima.

U [1], korištena je potpuno konvolucijska mreža slična U-Netu kojeg sam ja koristio, a u [8] korišten je, između ostalog, U-Net s gusto povezanim blokovima. Ti modeli bili su mi zanimljivi jer koriste slične arhitekture modela kao i ja.

Tablica 6.2: Rezultati 2D modela. U prvom dijelu tablice nalaze se moji rezultati, a u drugom dijelu tablice rezultati drugih radova na istom skupu podataka.

Model	Srednji Diceov koeficijent		
	Cijeli tumor	Jezgra tumora	Invazivna jezgra
U-Net	0.8668	0.7776	0.7210
SwiftNet-RN18	0.8790	0.8184	0.6953
Ladder DenseNet121	0.8797	0.8274	0.6910
FCNN [1]	0.79	0.65	0.63
Dense U-Net [8]	0.86	0.72	0.68

Zanimljivo je da su se SwiftNet-RN18 i DenseNet121 s ljestvičastim naduzorkovanjem pokazali boljima u segmentaciji cijelog tumora i jezgre tumora u odnosu na U-Net, iako su razvijeni za korištenje u svrhu analize scena vožnje. Daljnje poboljšanje točnosti moglo bi se ostvariti pronalaskom druge kombinacije uvećanja podataka. Postprocesiranje izlaza također se nudi kao opcija. Jedna mogućnost je korištenje 3D analize povezanih komponenata na izlazima i odbacivanje oznaka koje ne uspiju prijeći graničnu vrijednost [1].



Slika 6.1: Primjeri izlaza svakog od 2D modela i ispravne segmentacijske mape.

6.3. 3D modeli

Modeli koji su dali rezultate predstavljene u ovom odjeljku predviđali su segmentacijske mape za cijeli volumen. Pokušao sam utrenirati 3D adaptaciju modela ResNet18 na svojem skupu podataka. Nakon što sam utrenirao taj model, pokušao sam dodati i SPP modul.

6.3.1. Validiranje modela ResNet18 s integriranom normalizacijom po grupi i nelinearnom aktivacijom

Zbog memorijskih zahtjeva korištenih 3D modela, koristio sam integriranu normalizaciju po grupi i nelinearnu aktivaciju, to jest ABN, jer je to smanjilo memorijsko zauzeće i omogućilo mi da modele treniram s većim grupama podataka. Zbog načina na koji je izveden ABN, morao sam koristiti predaktivacijski rezidualni model. Da bih se uvjerio da ABN funkcionira i daje usporedivu točnost, pokušao sam utrenirati predaktivacijski ResNet18 za klasifikaciju na skupu podataka CIFAR-10. Kako se radi o modelu za klasifikaciju, ne za semantičku segmentaciju, izlazi posljednjeg sloja propuštaju se kroz potpuno povezani sloj. Također nema potrebe za primjenom SPP modula ili ikakvog oblika naduzorkovanja budući da je izlaz mreže skalar. Također nisam koristio parametre predtrenirane na ImageNet skupu podataka. Rezultati validiranja modela prikazani su u tablici 6.3.

Tablica 6.3: Rezultati validiranja predaktivacijskog modela ResNet18 s integriranom normalizacijom po grupi i nelinearnom aktivacijom na skupu podataka CIFAR-10. Kosinusno kaljenje i stepenice predstavljaju dva načina smanjivanja koraka učenja. Pritom, za stepenice sam korak učenja smanjio za faktor 10 nakon 100 i 150 epoha ako je model treniran 200 epoha, a ako je model treniran 350 epoha, onda sam korak učenja smanjio za faktor 10 nakon 150 i 250 epoha.

Optimizator	Broj epoha	Veličina grupe	Točnost (%)	
			Kosinusno kaljenje	Stepenice
Adam	350	128	0.9375	0.9380
		50	0.9391	0.9434
	200	128	0.9366	0.9348
		50	0.9397	0.9415
SGD sa zaletom	350	128	0.9013	0.9500
		50	0.1	0.1
	200	128	0.9094	0.9383
		50	0.8897	0.9440

Uspio sam utrenirati predaktivacijski ResNet18 tako da daje zadovoljavajuće rezultate na skupu podataka CIFAR-10, no performanse modela jako su ovisile o hiperparametrima korištenima prilikom treniranja. Najbolje rezultate ostvario sam kada sam koristio stohastički gradijentni spust sa zaletom na 350 epoha, s veličinom grupe 128 i stepeničastim smanjivanjem koraka učenja.

Zanimljiva je značajna razlika koju su činili načini smanjivanja koraka učenja. Koristio sam kosinusno kaljenje i stepeničasto smanjivanje koraka učenja. Pritom, za stepenice sam korak učenja smanjivao za faktor 10 nakon 100 i 150 epoha ako sam model trenirao 200 epoha, a ako sam model trenirao 350 epoha, onda sam korak učenja smanjio za faktor 10 nakon 150 i 250 epoha. Kada sam koristio optimizator Adam, način smanjivanja koraka učenja nije činio gotovo nikakvu razliku. Kada sam koristio stohastički gradijentni spust sa zaletom, kosinusno kaljenje davalo je rezultate koji su bili i do 5 postotnih bodova lošiji u odnosu na stepeničasto smanjivanje koraka učenja.

6.3.2. Dobiveni rezultati i usporedba s drugim radovima

U tablici 6.4 navedene su srednje vrijednosti Diceovih koeficijenata dobivene na skupu za testiranje. U prvom dijelu tablice naveo sam rezultate svojih modela, a u drugom dijelu tablice rezultate drugih radova koji su se bavili istim skupom podataka i također učili na čitavim volumenima, ne na neovisnim kriškama.

Tablica 6.4: Rezultati 3D modela. U prvom dijelu tablice nalaze se moji rezultati, a u drugom dijelu tablice rezultati drugih radova na istom skupu podataka.

Model	Diceov koeficijent		
	Cijeli tumor	Jezgra tumora	Invazivna jezgra
3D ResNet18	0.1087	0.0572	0.0197
3D ResNet18 + SPP	0.0183	0.0064	0
3D U-Net [17]	0.82	0.67	0.60
Ansambl 10 FCNN + VAE [14]	0.8839	0.8154	0.7664

Modeli koje sam koristio daju višestruko lošije rezultate od 2D modela. Pretpostavljam da 3D adaptacija ResNeta jednostavno nije dobro rješenje za segmentaciju glioma iz snimki mozga. U [17] korištena je 3D adaptacija U-Neta za učenje. Pritom se učenje odvija u dva koraka, u prvom koraku se iz volumena niske rezolucije izdvaja područje u kojem se nalazi tumor, a u drugom koraku se segmentira tumor iz samo onog dijela volumena koji sadrži tumor. Čak i ovaj pristup daje lošije rezultate od gotovo svih modela koji koriste samo neovisne 2D kriške. Modeli koji uzimaju 3D slike puno su veći i računalno ih je zahtjevno trenirati te očito zahtijevaju sofisticiranije pristupe treniranju. Tome u prilog govori i [14] gdje je korišten ansambl 10 potpuno konvolucijskih modela s koderom, dekoderom i varijacijskim autoenkoderom kao dodatnom metodom regularizacije. Taj model daje bolje rezultate od viđenih 2D

modela, no očigledno je bilo puno teže trenirati ga. Korišteni su manji isječci slike i model je treniran na NVIDIA Tesla V100 grafičkom procesoru s 32 GB memorije te je treniranje jednog modela trajalo 2 dana.

7. Zaključak

Ovaj rad istražuje primjenu dubokog učenja za analizu medicinskih slika, specifično za semantičku segmentaciju novotvorina u slikama magnetske rezonancije mozga. Novotvorine koje segmentiram u ovom radu su gliomi. Kao neke od najsmrtonosnijih tvorevina u mozgu, zahtijevaju često praćenje, što uključuje i ručnu segmentaciju tvorevina. Takav način segmentiranja vrlo je vremenski intenzivan i kada bismo našli način kako automatizirati segmentaciju, mogli bismo u velikoj mjeri rasteretiti zdravstveni sustav i olakšati liječenje pacijenata.

U ovom radu opisani su osnovni koncepti koje je potrebno razumjeti da bismo mogli oblikovati modele koje bismo mogli koristiti za semantičku segmentaciju tumora. Modele sam trenirao na podatkovnom skupu. On sadržava snimke mozgova u 4 načina snimanja i segmentacijsku mapu za svaki mozak.

Za segmentaciju tumora koristio sam 3 osnovne arhitekture modela: U-Net, SwiftNet i DenseNet121 s ljestvičastim naduzorkovanjem. Spomenute modele koristio sam za segmentaciju tumora iz individualnih krišaka snimki mozga. Također sam koristio i 3D adaptaciju modela ResNet18 kao pokušaj učenja na čitavim volumenima.

Koristeći modele za segmentaciju tumora iz krišaka volumena ostvario sam dobre rezultate u usporedbi sa sličnim pristupima iz literature. Iako su SwiftNet i DenseNet121 s ljestvičastim naduzorkovanjem obično korišteni za segmentaciju scena vožnje, daju dobre rezultate na problemu kojim se bavi ovaj rad. Dodatna poboljšanja točnosti mogla bi se postići drukčijim načinom uvećanja podataka ili inovativnim načinima postprocesiranja izlaza iz modela. Osim toga, pokušao sam utrenirati 3D adaptaciju modela ResNet18 za segmentaciju tumora iz čitavih volumena, no takav model puno je teže utrenirati i njime nisam uspio postići zadovoljavajuće rezultate.

LITERATURA

[1] Varghese Alex, Mohammed Safwan, i Ganapathy Krishnamurthi. Automatic Segmentation and Overall Survival Prediction in Gliomas using Fully Convolutional Neural Network and Texture Analysis. *arXiv e-prints*, art. arXiv:1712.02066, Prosinac 2017.

[2] S. Bakas et al. Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. *Scientific Data*, 2017.

[3] S. Bakas et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. 2019.

[4] Ian Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, art. arXiv:1512.03385, Prosinac 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Identity Mappings in Deep Residual Networks. *arXiv e-prints*, art. arXiv:1603.05027, Ožujak 2016.

[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, i Kilian Q. Weinberger. Densely Connected Convolutional Networks. *arXiv e-prints*, art. arXiv:1608.06993, Kolovoz 2016.

[8] Geena Kim. Brain tumor segmentation using deep fully convolutional neural networks. U Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Bjoern Menze, i Mauricio Reyes, urednici, *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, stranice 344–357, Cham, 2018. Springer International Publishing. ISBN 978-3-319-75238-9.

[9] Diederik P. Kingma i Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, Prosinac 2014.

[10] Ivan Krešo, Josip Krapac, i Siniša Šegvić. Efficient Ladder-style DenseNets for Semantic Segmentation of Large Images. *arXiv e-prints*, art. arXiv:1905.05661, Svibanj 2019.

[11] Elizabeth A. Maher et al. Malignant glioma: genetics and biology of a grave matter. *Genes Development*, 15(11):1311–1333, 2001. doi: 10.1101/gad.891601. URL `http://genesdev.cshlp.org/content/15/11/1311.short`.

[12] W.S. McCulloch i W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[13] B.H. Menze et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Trans Med Imaging*, 2015.

[14] Andriy Myronenko. 3D MRI brain tumor segmentation using autoencoder regularization. *arXiv e-prints*, art. arXiv:1810.11654, Listopad 2018.

[15] Marin Oršić, Ivan Krešo, Petra Bevandić, i Siniša Šegvić. In Defense of Pre-trained ImageNet Architectures for Real-time Semantic Segmentation of Road-driving Images. *arXiv e-prints*, art. arXiv:1903.08469, Ožujak 2019.

[16] Marin Oršić i Siniša Šegvić. Efficient semantic segmentation with pyramidal fusion. *Pattern Recognition*, 110:107611, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107611. URL `https://www.sciencedirect.com/science/article/pii/S0031320320304143`.

[17] R. G. Rodríguez Colmeiro, C. A. Verrastro, i T. Grosges. Multimodal brain tumor segmentation using 3d convolutional networks. U Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Bjoern Menze, i Mauricio Reyes, urednici, *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, stranice 226–240, Cham, 2018. Springer International Publishing. ISBN 978-3-319-75238-9.

[18] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*, art. arXiv:1505.04597, Svibanj 2015.

[19] Samuel Rota Bulò, Lorenzo Porzi, i Peter Kontschieder. In-Place Activated BatchNorm for Memory-Optimized Training of DNNs. *arXiv e-prints*, art. arXiv:1712.02616, Prosinac 2017.

[20] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, i Aleksander Madry. How Does Batch Normalization Help Optimization? *arXiv e-prints*, art. arXiv:1805.11604, Svibanj 2018.

[21] Eli Stevens, Luca Antiga, i Thomas Viehmann. *Deep Learning with PyTorch*. Manning Publications Co., 2020. `https://pytorch.org/assets/deep-learning/Deep-Learning-with-PyTorch.pdf`.

[22] Aston Zhang, Zachary C. Lipton, Mu Li, i Alexander J. Smola. *Dive into Deep Learning*. 2020. `https://d2l.ai`.

[23] Marko Čupić. Umjetna inteligencija - umjetne neuronske mreže. `http://java.zemris.fer.hr/nastava/ui/ann/ann-20180604.pdf`, 2016.

[24] Siniša Šegvić. Neslužbene stranice predmeta duboko učenje. `http://www.zemris.fer.hr/~ssegvic/du/`. Pristupljeno: 2. lipnja 2021.

[25] Jan Šnajder i Bojana Dalbelo Bašić. Strojno učenje. `https://www.fer.unizg.hr/_download/repository/StrojnoUcenje.pdf`, 2014.

Semantička segmentacija novotvorina u slikama magnetske rezonancije

Sažetak

U ovom radu primijenjene su metode dubokog učenja u svrhu semantičke segmentacije glioma iz snimki magnetske rezonancije mozga. Napravljen je kratki pregled osnovnih koncepata dubokog učenja i predstavljene su tri arhitekture modela koje sam koristio za segmentaciju: U-Net, SwiftNet i DenseNet121 s ljestvičastim naduzorkovanjem. Modeli su korišteni za segmentaciju tumora u kriškama snimki mozga. Uz to, korištena je i 3D adaptacija modela ResNet18 za segmentaciju u volumenima mozga. Napravljena je usporedba dobivenih rezultata i rezultata iz literature.

Ključne riječi: analiza medicinskih snimki, gliom, neuronske mreže, duboko učenje, konvolucijske neuronske mreže, semantička segmentacija, U-Net, SwiftNet, DenseNet, ljestvičasto naduzorkovanje

Semantic segmentation of neoplasms in magnetic resonance images

Abstract

This paper applies deep learning to the problem of semantic segmentation of glioma in MR brain scans. This paper offers a short overview of basic concepts of deep learning and presents three model architectures that were used: U-Net, SwiftNet and DenseNet121 with ladder-style upsampling. The models were used for semantic segmentation of glioma in individual slices of the brain scan volumes. A 3D adaptation of ResNet18 was used for segmentation in entire brain scan volumes. The paper also compares results obtained using the aforementioned models with results from other papers on the same dataset.

Keywords: medical image analysis, glioma, neural networks, deep learning, convolutional neural networks, semantic segmentation, U-Net, SwiftNet, DenseNet, ladder-style upsampling