SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6120

DETEKCIJA ZUBA U ORTOPANTOMOGRAMIMA

Miljenko Šuflaj

Zagreb, lipanj 2019.

Zahvala

Zahvaljujem mentoru, **izv. prof. dr. sc. Marku Subašiću**, na podršci usprkos putu u nepoznato i svim problemčićima koji bi mnoge obeshrabrili.

Također zahvaljujem i kolegici **Ani Andrašek** koja je priložila svoj ortopantomogram različite rezolucije te znatno pomogla u otklanjanju iznimno velikog problema kod dobivanja rezultata; bez njenog ortopantomograma ovaj rad bi dugo čekao na otklanjanje pogreške.

Konačno, zahvaljujem svim **prijateljima** i **obitelji** na podršci kroz studij općenito bez kojih nikada ne bih došao ovoliko daleko.

Sadržaj

Zahvala 1
Sadržaj 2
Uvod 4
Umjetne neuronske mreže 5
Konvolucijske neuronske mreže 6
Konvolucijski sloj 7
ReLU sloj 9
Sloj združivanja 9
Poptuno povezani sloj 11
Sloj gubitka 11
Razmatrane arhitekture neuronskih mreža 12
Jednoprolazne i dvoprolazne neuronske mreže 12
Arhitektura RetinaNet 14
Komponente 15
Izlučivanje značajki 15
Razredbena konvolucijska neuronska mreža 16
Pronalazna konvolucijska neuronska mreža 17
Princip rada 18
Računanje gubitka 19
Razredbeni gubitak 19
Pronalazni gubitak 20
Zaključivanje 21
Postupak pripreme i interpretacije podataka 22
Rezultati 26
Princip ocjenjivanja točnosti mreže 33

Određivanja praga za veličinu slike uz očuvanje točnosti 38
Usporedba s drugim arhitekturama srodnog zadatka 41
Zaključak 43
LITERATURA 44
Sažetak 46
Summary 47

Uvod

Problem detekcije zuba u ortopantomogramima je naizgled jednostavan problem rješavaju li ga ljudi. Osobe koje nemaju kvalifikacije, tj. nisu stomatolozi mogu iznimno lagano pronaći zube te ih razrediti znaju li neke osnovne stvari o ljudskim zubima. Problemi nastaju kada trebao uočiti zube u slikama loše kvalitete ili uočiti zube loše kvalitete. U tom slučaju čak će i stručnjaci s vremena na vrijeme završiti s češkanjem po tjemenu. Kao i kod svakog problema, ljudi su vični tražiti spas u računalima koja u trenutku pisanja ovog rada posjeduju zaista zavidnu snagu obrade te su zasigurno izum koji je najviše pomogao razvoju znanosti u 21. stoljeću. Problem uočavanja zuba je problem čiji su ulazni parametri vizualni podražaji. Znanost još uvijek nije standardizirala pretvorbe vizualnih signala u jednoznačne tekstualne podatke, pa se stoga moramo baciti u granu neuroračunarstva, koja za jednu od podgrana svojata računalni vid. Problem detekcije zuba u području neuroračunarstva može se raščlaniti na 2 potproblema; problem pronalaska zuba (engl. *regression*) i problem razredbe zuba (engl. *classification*). Ovi problemi se uobičajeno rješavaju konvolucijskim neuronskim mrežama (engl. *convolutional neural networks)* koje imaju sposobnost učenja damo li im dovoljno veliki skup označenih podataka. U neuroračunarstvu postoje još mnogi pristupi koje bismo mogli iskoristiti za rješavanje ovih problema, no u vrijeme pisanja ovog rada, konvolucijske neuronske mreže su de facto standard rješavanja problema nalik ovome.

Usprkos tome što smo već rano odabrali tehniku rješavanja ovog problema, konvolucijske neuronske mreže su vrlo općenita tehnika, korištena za mnogo različitih tipova problema predočenih vizualnim podražajima, pa ćemo, nakon što objasnimo osnovne pojmove potrebne za usvajanje osnovnog znanja o konvolucijskim neuronskim mrežama pričati i o različitim inačicama i arhitekturama sustava za pronalazak i razredbu zuba.

Uvod u konvolucijske neuronske mreže

Konvolucijske neuronske mreže je jedna od mnogobrojnih inačica umjetnih neuronskih mreža (engl. *artificial neural networks*). Umjetne neuronske mreže su računalni sustavi koji oponašaju živčane sustave u mozgovima životinja. Specifičnost takvih sustava je što uče promatranjem primjera bez da su unaprijed uhodani u neki specifični postupak rješavanja zadanog problema. U pogledu računalnog vida, umjetne neuronske mreže uobičajeno uče promatranjem označenih objekata na foto ili videosadržaju dok se ne postigne zadovoljavajuća učestalost točnog zaključka za objekte koji se razmatraju [1].

Umjetne neuronske mreže

Općenito, umjetne neuronske mreže sastoje od mnoštva osnovnih jedinica koje nazivamo neuronima koji su analogni s neuronima u mozgu životinja. Svaka veza između neurona, nalik sinapsama, prenosi signale iz jednog neurona u drugi. Uobičajeno je da neuroni prenose realne brojeve, dok se izlaz neurona računa primjenom odabrane funkcije, obično nelinearne, uvrštavanjem sume ulaza. Veze između neurona nazivamo rubovi (engl. *edges*) te je uobičajeno da oni sadrže koeficijent koji podešava snagu emisije signala. Taj koeficijent se najčešće mijenja tijekom učenja. Osim toga, moguće je postaviti prag ispod kojega se signal neće poslati.

Neurone uobičajeno ne gledamo kao autonomne jedinice, već u pogledu slojeva. Slojevi se razlikuju po funkcionalnosti, oni na različite načine preoblikuju ulaze, a u osnovnom slučaju imamo 2 sloja; ulazni i izlazni sloj. Broj neurona u ulaznom sloju mora se poklapati s brojem značajki ulaza, dok se broj neurona u izlaznom sloju mora poklapati s brojem značajki izlaza. Konkretno, želimo li neku rastersku sliku veličine 32 x 32 razvrstati u slike mačke, psa, kornjače ili zeca, naša neuronska mreža imat će 1024 neurona u ulaznom sloju te 4 neurona u izlaznom sloju. Ograničenja osnovne arhitekture je što neuronska mreža sa samo ulaznim i izlaznim

slojem nema mogućnost rješavanja problema nelinearnih problema. Ovo je zato što je neuronska mreža tada samo reprezentacija jedne linearne kombinacije parametara iz ulaznog sloja koji se preslikavaju u 4 vrijednosti. Zbog toga se uvodi pojam skrivenih slojeva (engl. *hidden layers*).

Skriveni slojevi su svi slojevi između ulaznog i izlaznog sloja [2]. Koriste se kada je problem prekompleksan za jednu linearnu kombinaciju parametara. Dok bi nam, u računalnom vidu, mreža bez skrivenih slojeva mogla prepoznati neke vrlo primitivne uzorke, trebali bi jedan skriveni sloj za detekciju rubova. Dodavanjem dodatnih skrivenih slojeva možemo dobiti sofisticiranije uočavanje složenijih značajki. Međutim, dodavanje skrivenih slojeva nije nešto što treba raditi preko mjere. Dodavanjem skrivenih slojeva povećavamo složenost, ali i specifičnost neuronske mreže. Previše neurona može rezultirati preprilagođenošću (engl. *overfitting*). Isto tako, premalo neurona može rezultirati podprilagođenošću (engl. *underfitting*) [1].

Konvolucijske neuronske mreže

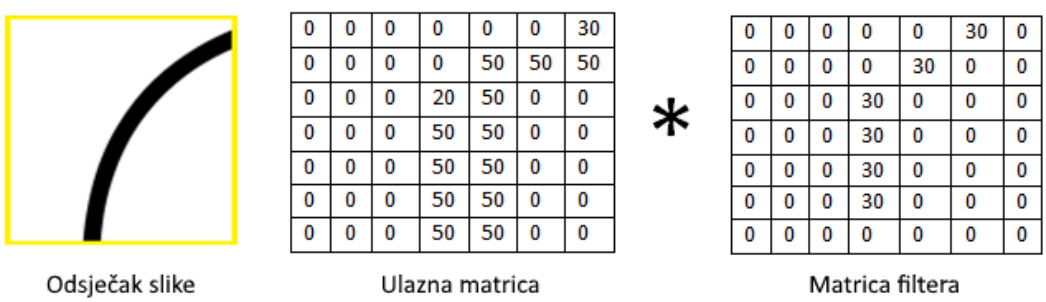
Konvolucijske neuronske mreže razlikuju se od tradicionalnih neuronskih mreža po tome što se tijekom učenja podešavaju vrijednosti tzv. filtera. Filteri su zapravo vektori realnih vrijednosti koji se učenjem podešavaju, a tijekom zaključivanja (engl. *inference*) konvoluiraju s ulaznim podacima. Filteri mogu biti različitih veličina, a u praksi su najuobičajeniji 1 x 1 i 3 x 3 filteri.

Konvolucijske neuronske mreže su prve neuronske mreže učenja s učiteljem (engl. *supervised learning*) koje su postigle rezultate usporedive s ljudima za probleme koji se njima rješavaju [3].

Klasično, konvolucijske neuronske mreže su razred dubokih neuronskih šalji-naprijed (engl. *feed forward*) mreža. Uobičajeno je da imaju 1 ili više potpuno povezanih konvolucijskih slojeva. Također je uobičajeno da se težine (engl. *weights*) dijele između slojeva i združivanje (engl. *pooling*), tj. združivanje po maksimumu (engl. *max pooling*).

Konvolucijski sloj

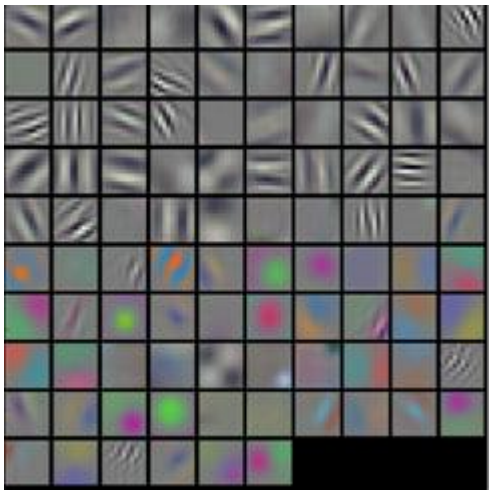
Konvolucijski sloj vrši konvoluciju određenog dvodimenzionalnog podatka s filterom konvolucijskog sloja. Učenjem mreže parametri filtera se podešavaju ne bi li dali što točniji zaključak o priloženom podatku iz baze za učenje. Konvolucijski filtri su obično matrice dimenzija N x N, te u praksi uglavnom govorimo o 1 x 1, 3 x 3, 5 x 5 filtrima. Ideja konvolucijskog sloja je preslikati dvodimenzionalne podatke, obično odsječak slike u jednodimenzionalnu vrijednost. Ovo se postiže konvolucijom matrica ulaza i filtera, tj. množenjem vrijednosti matrica po elementima (engl. *element-wise matrix multiplication*).



Slika 1: Rad konvolucijskog sloja [4]

Na slici iznad izlaz glasi $50 \cdot 30 + 50 \cdot 30 + 50 \cdot 30 + 50 \cdot 30 + 20 \cdot 30 = 6600$

Korištenjem različitih filtera možemo dobiti različitu funkcionalnost konvolucijskih slojeva, tj. oni će zapažati različite stvari (npr. lijeve rubove, desne rubove, određene vrste krivulja itd.). Filteri vuku analogiju različitih vrsta češljeva, jer oni „češljaju“ ulazne podatke u oblike koji bolje ili lošije prikazuju neke značajke danih podataka. Moguće ih je vizualizirati kao sitne sličice:



Slika 2: Vizualizacija filtera [5]

Neuroni u konvolucijskom sloju posloženi su u 3 dimenzije; širini, visinu i dubinu. Svaki sloj povezan je samo sa djelićem sloja prethodnika. Taj djelić zovemo polje osjeta (engl. *receptive field*). Udruživanjem raznih konvolucijskih slojeva, nekih lokalno povezanih, nekih globalno povezanih, dobivamo konvolucijsku neuronsku mrežu.

Razlog zašto sada odjednom imamo lokalno, a ne globalno povezivanje je jer želimo da naši neuroni stvaraju najsnažniji odziv na prostorno lokalne uzorke. Slaganjem mnogo lokalno povezanih slojeva dobivamo nelinearne filtere koji veću sliku rekonstruiraju slaganjem manjih dijelova.

Finalno, prije smo spomenuli da se filtri ponavljaju jer se svakom neuronu u istom sloju pridružuju iste težine, tj. isti filtri. Ovo rezultira time da svaki neuron u takvom sloju biva pobuđen istim značajkama za svaku pobudu u svojem polju osjeta. Ovo omogućuje da uočavanje značajki ne ovisi o lokaciji na slici, tj. da se svaka značajka koju želimo uočiti može uočiti gdje god se nalazila na slici.

Ova tri svojstva omogućavaju bolju generalizaciju. Dodatno, dijeljenje težina smanjuje snagu obrade te broj parametara koje mreže treba podesiti, što smanjuje zahtjevnost cijelog algoritma te omogućuju treniranje većih mreža.

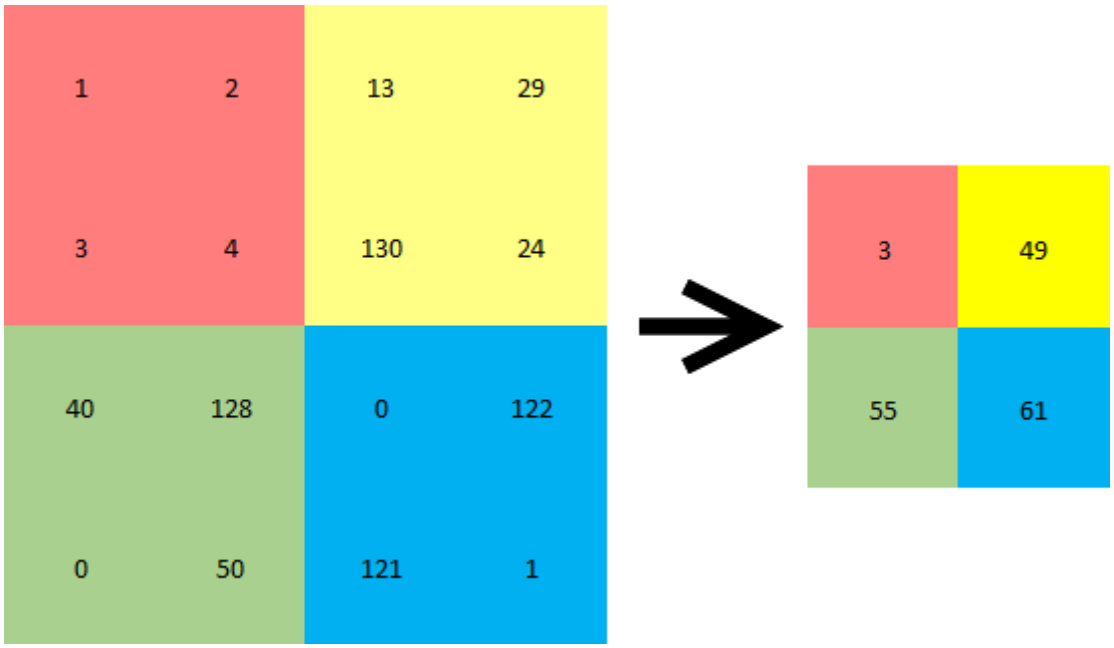
ReLU sloj

ReLU (engl. *Rectified Linear Unit*) je funkcija oblika $\max(0, x)$. Ova funkcija, kao što se može vidjeti izdaleka, ruši negativne vrijednosti u 0, dok čuva pozitivne vrijednosti onakvim kakve jesu. ReLU sloj u konvolucijskoj neuronskoj mreži povećava nelinearna svojstva funkcije odluke (engl. decision function), ali i cijele mreže bez izmjene podataka polja osjeta konvolucijskog sloja.

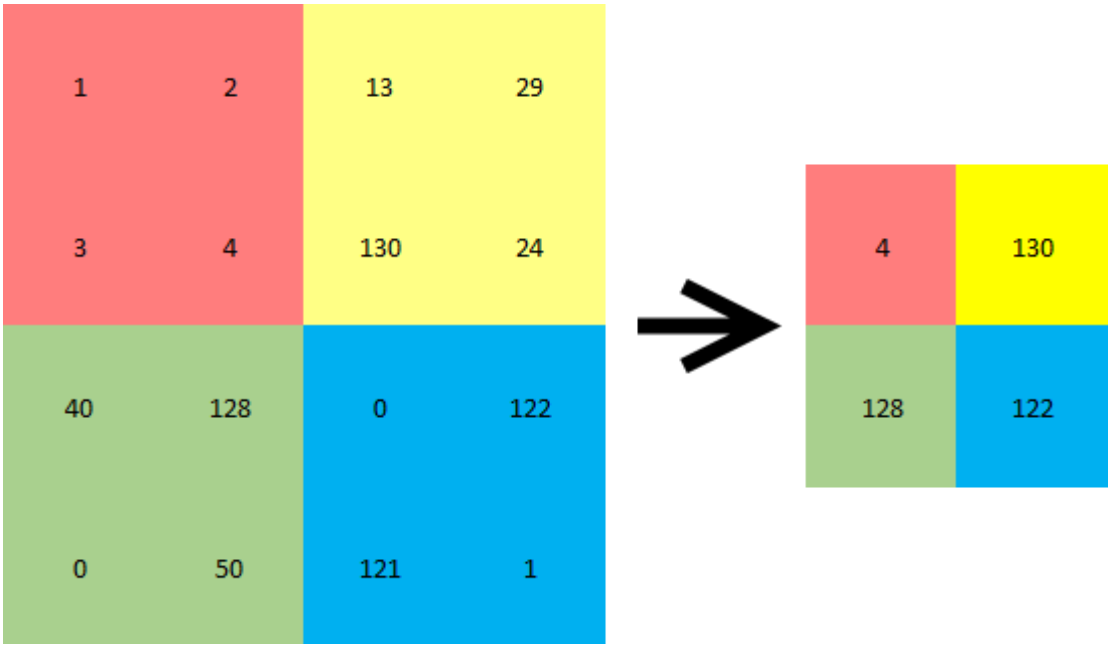
Za sloj iste funkcionalnosti kao ReLU postoje i druge poznate funkcije kao što su hiperbolni tangens ($\tanh(x)$) i sigmoid ($\frac{1}{1+e^{-x}}$) [6]. Razlog zašto konvolucijske neuronske mreže uglavnom biraju ReLU za funkciju uključenja (engl. *activation function*) je zato što se postiže nekoliko puta brže učenje bez značajnih kazni u smislu općenite preciznosti mreže.

Sloj združivanja

Slojevi združivanja vrše združivanje na podacima. Združivanje je jedna od metoda nelinearnog poduzorkovanja (engl. *downsampling*), čime možemo smanjiti dimenzionalnost podataka. Postoji nekoliko funkcija kojima možemo postići združivanje, npr. združivanje po prosjeku i združivanje po maksimumu. Iako bi laik rekao da je združivanje po prosjeku logičniji izbor za funkciju združivanja, u praksi se najviše koristi združivanje po maksimumu. Kada smanjujemo podatke, bitnije nam je sačuvati bitne podatke sa slike. Unaprijed pretpostavimo da je podatak bitniji što mu je vrijednost kojom je iskazan veća. Kad bi se radilo o poduzorkovanju slike s vizualnog aspekta, bolje bi bilo koristiti združivanje po prosjeku, no u slučaju konvolucijskih neuronskih mreža, pristup sa združivanjem po maksimumu pokazao se boljim. Kao parametar pri združivanje koristi se veličina klizećeg prozora (engl. *sliding window size*), dvodimenzionalna vrijednost koja je uobičajeno 2x2, te korak (engl. *stride*), što je udaljenost koju će u sljedećoj iteraciji združivanja prozor prijeći kako bi došao do sljedećeg skupa vrijednosti koje treba združiti.



Slika 3: Združivanje s prosjekom, 2x2 prozor, korak 2



Slika 4: Združivanje s maksimumom, 2x2 prozor, korak 2

Motivacija kod združivanja nije samo jedna. S jedne strane, Ahilova peta umjetnih neuronskih mreža je dugo vrijeme treniranja. Ono može biti takvo zbog veličine ulaznih podataka, broja iteracija ili epoha, te veličine same mreže. Cilj nam je

smanjiti podatke koliko god možemo, a opet sačuvati informaciju koja se nalazi u tim podacima koliko god dobro možemo.

Poptuno povezani sloj

Nakon slaganja nekoliko konvolucijskih i slojeva združivanja, konvolucijska mreža sadržavat će jedan ili više potpuno povezanih slojeva. Potpuno povezani slojevi su slojevi neurona koji primaju sve izlaze prethodnog sloja. U matematici bi potpuno povezanim slojevima bilo analogno matrično množenje, samo što u ovom slučaju možemo imati neki parametar koji će nam određivati pomak (engl. *bias*), realan broj koji ćemo zbrojiti i time dobiti izlaz neurona.

Sloj gubitka

Jedan od bitnih slojeva u konvolucijskoj neuronskoj mreži je sloj gubitka. Ovaj sloj je zaslužan za računanje pogreške u zaključivanje neke konvolucijske neuronske mreže. Kako je funkcija gubitka (engl. *loss function*) proizvoljno odabrana funkcija, nama je u interesu da ona što vjernije prikazuje inverz točnosti za problem koji pokušavamo riješiti našom konvolucijskom neuronskom mrežom [7]. Ovisno o rasponu vrijednosti koje nam vraća ostatak mreže, prikladno je odabrati Softmax gubitak (koristimo kad predviđamo K klasa s međusobno isključivim vjerojatnostima), Sigmoidalnu križnu entropiju (engl. *sigmoid cross-entropy*) kad predviđamo K klasa, svaka vjerojatnosti u rasponu [0, 1], te Euklidski gubitak (engl. *Euclidean loss*) kad su nam vjerojatnosti za svaku od K klasa realni brojevi.

Razmatrane arhitekture neuronskih mreža

Prilikom rješavanja problema uočavanja zuba u ortopantomogramima, zbog toga što je računalni vid kao grana relativno razvijeno područje, mogli smo birati između puno vrsta neuronskih mreža. Na ovu temu postoji već nekoliko radova u kojem je korišten pristup dvoprolaznih neuronskih mreža (engl. *two pass neural network*), specifično arhitekture Faster R-CNN [8]. U tom radu [9] uspoređivana je arhitektura Faster R-CNN s drugim jednoprolaznim neuronskim mrežama (engl. *single pass neural network*), kao što je YOLO [10].

Jednoprolazne i dvoprolazne neuronske mreže

Razlika između jednoprolaznih i dvoprolaznih neuronskih mreža nalazi se u broju prolaza ili faza koje neuronska mreža tijekom učenja prođe. Specifično, ono što je posebno kod arhitekture Faster R-CNN je da ona koristi mrežu za preporuku područja (engl. *Region Proposal Network*) kako bi prvo izolirala interesantna područja na slici, a tek potom išla uočavati objekte te ih klasificirati.

Ovakav pristup je potreban jer u općenitom slučaju na slici se može nalaziti puno pozadine. Ovo će dovesti do nejednakosti klasa, tj. s obzirom na to da je na slikama iz baze za učenje najprisutnija pozadina, mreže će zanemarivati objekte koje treba uočiti i označiti. Ovo je rezultiralo redundancijom i lošom općenitom točnošću, pa se tvorci arhitekture Faster R-CNN doskočili potencijalnim uravnoteženjem primjeraka pozadine i razreda objekata koje želimo detektirati.

Jednoprolazne neuronske mreže ovo nisu radile, već su se jednostavno, nakon izlučivanja značajki, bacile na posao treniranja mreže. S jedne strane, zbog toga što su prilikom učenja radile manje posla, mreže su brže učile, no njihova je točnost bila manja, što zbog nejednakosti klasa, što zbog drugih problema, npr. razne skale objekata.

Donedavno, dvoprolazne neuronske mreže bile su de facto standard zbog neusporedivo bolje točnosti uz vrlo male kazne na performanse.

Kad smo krenuli rješavati problem uočavanja zuba na ortopantomografima, moje reference bile su upravo mreže bazirane na Faster R-CNN-u. Inicijalno, plan je bio istestirati takvu mrežu, no kada sam preuzimao gotovu implementaciju kao relativno zelen novak u području računalnog vida, poruka da je Faster R-CNN zastario pojavila mi se na ekranu uz poveznicu na navodno bolju arhitekturu.

Arhitektura o kojoj pričamo naziva se RetinaNet [11]. Tijekom istraživanja postalo je jasno da je upravo RetinaNet arhitektura, već tada dosta proslavljena arhitektura nešto čime bi se trebalo pozabaviti.

Arhitektura RetinaNet

Prva od boljki kojom se ponosi arhitektura RetinaNet je da je to jednoprolazna neuronska mreža. Drugo iznenađenje kod mreže jest računanje pogreške. Radi se o nečemu što su originalni autori nazvali žarišni gubitak (engl. *focal loss*), a funkcija gubitka glasi:

$$L(p) = -\alpha(1-p)^{\gamma}\log(p) \qquad (1.1)$$

Analizirajmo malo priloženu jednadžbu. Na prvi pogled, ovo izgleda kao križna entropija (engl. cross entropy). Razlika je u novim hiperparametrima koje smo uveli, točnije α i γ.

Prvi hiperparametar, α, služi kao broj koji će skalirati ukupan gubitak, te je uobičajeno u rasponu [0, 1]. U svojem radu, autori preporučuju da se α postavi na inverz frekvencije neke klase.

Drugi hiperparametar, γ, služi kao faktor kojim će se izraz u zagradi, $(1-p)$, potencirati, te je to uobičajeno broj u rasponu $[0, \infty\rangle$. Ako analiziramo malo taj izraz, primijetit ćemo da je on uvijek u rasponu [0, 1], tako da će cijeli taj faktor konvergirati u 0 povećavanjem drugog hiperparametra. S obzirom na to da zapravo potenciramo vjerojatnost da detektirani objekt nije onaj koji je klasificiran, možemo primijetiti da za γ koji je 1 ili više možemo dobiti faktor koji će smanjivati doprinose gubitka što je klasifikator sigurniji oko zaključka o klasi. Ovo će u prijevodu značiti da će klasifikator biti stroži oko krivih pretpostavki malih sigurnosti, tj. da će se usredotočiti na optimiziranje neuronske mreže za raspoznavanje težih primjera. Uobičajeno je da se za hiperparametar γ uzima vrijednost 2, a ako mu damo vrijednost 0, žarišni gubitak postaje križna entropija.

Naposljetku, parametar p, koji je ujedno i parametar funkcije gubitka je sigurnost s kojom je donesena odluka, a to je uobičajeno vrijednost u intervalu [0, 1].

Ovakvim pristupom želi se boriti protiv nejednakosti razreda. Razredi koji su vrlo zastupljeni će zbog velike sigurnosti manje doprinositi ukupnom gubitku. Mreža će se stoga posvetiti težim, manje zastupljenijim razredima, pa bi si mogli dozvoliti ukidanje procesa pronalaska interesantnih područja, tj. totalno eliminirati mrežu za preporuku područja iz cijele priče.

Komponente

Usprkos upravo navedenim značajkama arhitekture RetinaNet, ona nije obična konvolucijska neuronska mreža uz navedene značajke. Zapravo, radi se u skupu komponenti koje rješavaju neke od problema široko prisutnih u računalnom vidu. Općenito, RetinaNet se sastoji od jedne kičmene mreže (engl. *backbone network*) koja izlučuje značajke te vraća konvolucijsku mapu značajki (engl. *convolutional feature map*), te je ta mreža implementirana kao gotovo rješenje. Zatim postoje dvije jednostavne konvolucijske neuronske mreže koje redom obavljaju razredbu nad izlazom prve mreže, te pronalazak okvira graničnika (engl. *bounding box*).

Izlučivanje značajki

Izlučivanje značajki radi se korištenjem mreže piramide značajki (engl. *Feature Pyramid Network*) [12]. Mreža piramide značajki je uobičajena konvolucijska neuronska mreža puta od gore prema dolje (engl. *top-down pathway*). Ono što radimo tijekom izlučivanja značajki je da konvolucijskim filtrima smanjujemo dimenzije slike. Iako konstruiramo 7 slojeva iz originalne slike, za konstrukciju mapa značajki (engl. *feature map*) koristit ćemo slojeve C_3 do C_7. Brojevi ispod slova označavaju smanjenje dimenzija originalne slike za 2^3 do 2^7 puta. Nakon generacije poduzorkovanih slojeva, konvoluiramo svaki sloj 1x1 filtrom, generirajući značajke M_7. Značajke M_7 generirat će našu najnižu razinu piramide, P_7. Nakon toga, spajanjem slojeva M_7 i 1x1 konvolucije C_6 dobivamo sloj P_6, i tako redom do P_3. Spajanje slojeva se vrši na način da se manji sloj preuzorkuje (engl. *upsampling*) funkcijom najbližeg susjeda (engl. *nearest neighbour*), vrijednosti se po elementima

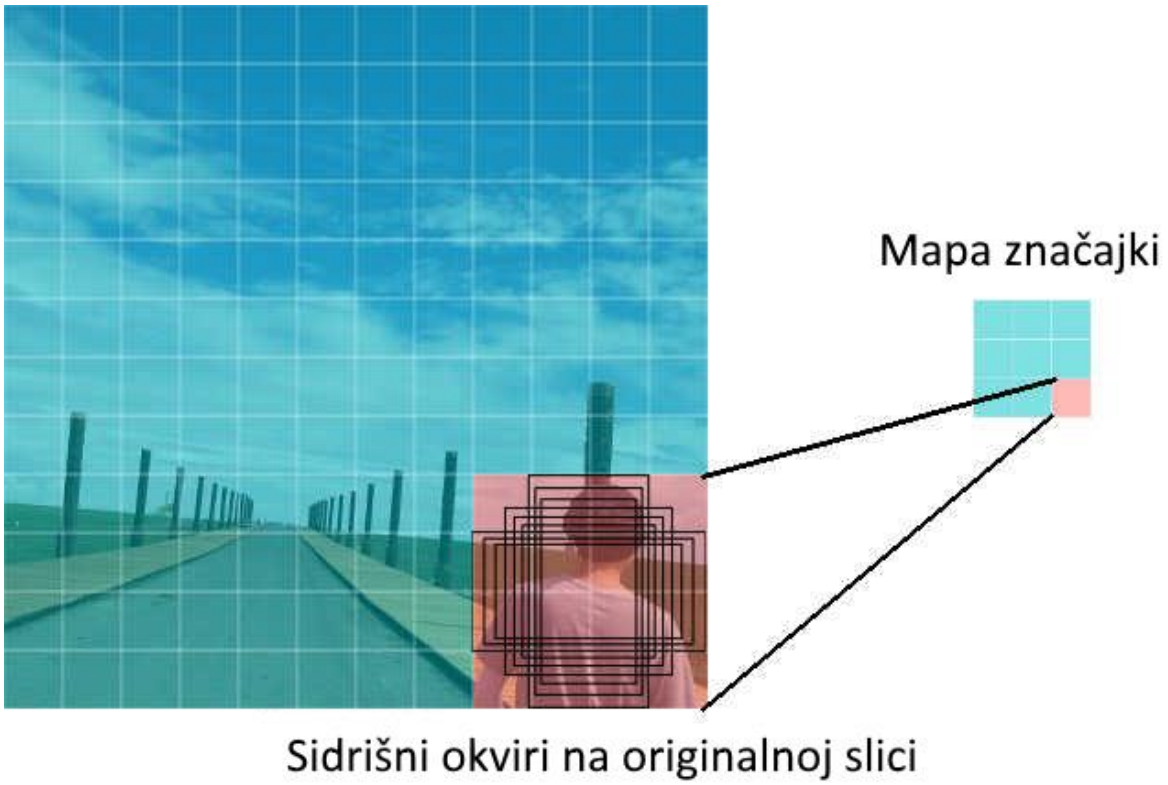
zbroje, te se svaki sloj koji nije onaj najniže rezolucije, koji je nastao bez spajanja, još jednom pročešlja 3x3 konvolucijom kako bi se smanjio *aliasing*. Ovom generacijom slojeva smo stvorili mape značajki koje mogu predviđati objekte različitih razmjera, a po testovima znatno ubrzavaju proces izlučivanja značajki te znatno povećavaju točnost neuronske mreže.

Iako se razredbene i pronalazne mreže u literaturi obično nazivaju podmrežama, u radu će se nadalje referencirati kao samo razredbene ili pronalazne konvolucijske neuronske mreže.

Razredbena konvolucijska neuronska mreža

Razredbu u RetinaNet arhitekturi obavlja potpuno povezana konvolucijska mreža (engl. *fully connected network*), priključena na svaki sloj značajki izlučenih pomoću mreže piramide značajki. Osnovna implementacija se sastoji od 4 sloja konvolucijska sloja dimenzija 3 x 3, te sadrži 256 filtera koji se aktiviraju ReLU funkcijom uključenja, koji vraćaju izlaz na posljednji konvolucijski sloj, također dimenzija 3 x 3 koji ovog puta sadrži K x A filtera, gdje je K broj razreda u koje ćemo svrstati neki objekt, dok je A broj sidrišnih okvira (engl. *anchor box*), čiji izlaz će se uključivati funkcijom sigmoida. Svaka razina ove mreže ima dijeljene težine. U broj razreda ne ulazi i implicitni razred pozadine, a sidrišni okviri su koncept koji smo uveli za ovu mrežu.

Sidrišni okviri su područja originalne slike različitih omjera širine i visine koje koristimo za označavanje jedne ćelije izlaza mreže piramide značajki. Zamislimo da je izlaz naše mreže piramide značajki N x M dimenzija. Tada ćemo za svaku ćeliju (N^2 njih) definirati A sidrišnih okvira. U slučaju RetinaNet arhitekture, A = 9, te su omjeri 1:1, 1:2 i 2:1. Za svaki omjer imamo 3 sidrišna okvira različitih veličina čija su središta u središtu područja slike koje analiziramo.



Slika 5: Primjer sidrišnih okvira [12]

Svaki sidrišni okvir u sebi sadrži vjerojatnosti za K razreda. Sidrišnim okvirima postižemo da uočeni predmet na slici može biti različitog oblika i veličine.

Izlaz razredbene konvolucijske neuronske mreže je oblika (W, H, KA), gdje W i H označavaju širinu i visinu ulazne mape značajki. Svaka kombinacija širine i visine na mapi značajki odgovara jednom odjeljku originalne slike, za koji ćemo onda imati KA parametara koji će nam govoriti o vjerojatnostima za svaki od K razreda u svakom od A sidrišnih okvira.

Pronalazna konvolucijska neuronska mreža

Pronalazak objekata vrši potpuno povezana konvolucijska neuronska mreža koja je gotovo identična razredbenoj, a jedina razlika se nalazi u zadnjem konvolucijskom sloja; naime, iako je također dimenzija 3 x 3, on sadrži 4A filtera. Stoga, izlazne dimenzije mape značajki je (W, H, 4A). Razlog zašto imamo 4A filtera je vrlo jednostavan; kod uočavanja objekata želimo označiti pomak od središta sidrišnog okvira, te veličinu uočenog objekta. Ovo činimo s 4 vrijednosti: x, y, w, h, te je zato

dimenzionalnost mape značajki (W, H, 4A). Semantika prve dvije dimenzionalnosti je jednaka kao i za razredbenu mrežu.

Princip rada

Princip rada ove arhitekture je, srećom, vrlo jednostavan. Prvo, ulazna slika se koristi kako bi se izgenerirale mape značajki koje zadržavaju semantički bitne značajke u nekoliko slojeva, te olakšavaju uočavanje objekata različitih veličina, omjera, osvjetljenja i sl. Ovo se radi mrežom piramide značajke, a u našem slučaju, za to koristimo mrežu ResNet50 [13].

Nakon što smo izgenerirali mape značajki, razredbena i pronalazna mreža uzima u obzir sve slojeve mapa značajki te na izlazu vraća mape značajki dimenzija (W, H, KA) i (W, H, 4A). Nakon zaključivanja, potrebno je izračunati gubitak koji na prvi pogled nije trivijalan postupak.

Kod računanja gubitka ideja je usporediti označene podatke sa zaključkom mreže te evaluirati koliko je zaključak mreže „promašio“ oznaku. U našem slučaju postoji velik broj zaključaka, pa prvo moramo odabrati one zaključke koje ćemo prihvatiti kao takve, a ostale odbaciti. S obzirom na to da se naši zaključci nalaze u tzv. izlaznim tenzorima (engl. *output tensors*), iz oznaka bi trebalo generirati ciljne tenzore (engl. *target tensors*). U općenitom slučaju, stvorit ćemo tenzore istih veličina kao i izlazni tenzori te na mjestima gdje smo u skupu za učenje označili neki predmet postavljamo vrijednosti oznaka u ciljni tenzor.

Kažemo da je sidrišni okvir odgovara okviru oznake (engl. *ground truth box*) ako je njihov presjek po uniji (engl. *intersection over union*) veći od 50%. Tada će se za sve klase za koje ovo vrijedi u ciljnom tenzoru postaviti vektor duljine K s jedinicama na mjestima koje predstavljaju razrede koji su uspješno spareni u ovom procesu, a 0 na svim ostalim mjestima. Ako se dogodi da za neki sidrišni okvir ne postoji presjek po uniji s okvirom oznake veći ili jednak 40%, taj sidrišni okviri tada smatramo pozadinom, te ga u ciljnom tenzoru interpretiramo kao vektor duljine K s vrijednosti 0 na svakom mjestu.

Međutim, postoji i treći slučaj – ako se dogodi da sidrišni okvir ima presjek po uniji između 40% i 50% za sve okvire oznake, smatramo da on nema para.

Računanje gubitka

Gubitak u RetinaNetu računa se kao suma dva člana; razredbenog gubitka (engl. *classification loss*) i pronalaznog gubitka (engl. *regression loss*). Možemo ga pisati kao:

$$L = L_{razredbe} + \lambda L_{pronalaska} \tag{1.2}$$

gdje λ označava hiperparametar s kojim se uravnotežuju razredbeni i pronalazni gubitci. Ovaj parametar je potreban s obzirom na to da se za gubitke koriste dvije bitno različite funkcije za računanje gubitka. Valja napomenuti da se u našem slučaju nije podešavao parametar λ, pa su gubitci bile neuravnoteženi. Srećom, ova arhitektura ne prioritizira napredak ovisno o iznosu pojedinog gubitka, pa smo izgubili isključivo na estetici konačnog iznosa gubitka.

Razredbeni gubitak

Kao što smo prethodno naglasili, jedan od aseva ove arhitekture je računanje razredbenog gubitka. Podsjetimo se, izraz glasi:

$$L_{razredbe}(p) = -\alpha(1-p)^{\gamma}\log(p) \tag{1.1}$$

Također se podsjetimo da ovom funkcijom želimo riješiti problem neuravnoteženosti razreda (engl. *class imbalance*) koji je veliko ograničenje detektora u praksi. Ovo se postiže parametrom γ koji usredotočuje mrežu na ispravljanje zaključivanja za

zaključke niskih vjerojatnosti, dok parametar α pokušava uravnotežiti utjecaj nekog mnogobrojnijeg razreda na ostale uravnoteživanjem doprinosa kod gubitka, te je on obično inverz frekvencije pojavljivanja razreda za koji se zaključivalo.

Pronalazni gubitak

Pronalazni gubitak je, srećom, otprije poznata funkcija gubitka. Naime, radi se o glatkom L1 gubitku (engl. *smooth L1 loss*) koji se može predočiti sa sljedećom funkcijom:

$$L_{pronalaska} = \sum_{j \in \{x,y,w,h\}} s_{L1}(P_j^i - T_j^i) \tag{1.3}$$

gdje se funkcija s_{L1} definira kao:

$$s_{L1}(x) = \begin{cases} 0.5x^2 \; za \; |x| < 1 \\ |x| - 0.5 \; za \; |x| \geq 1 \end{cases} \tag{1.4}$$

a parametri T_x^i, T_y^i, T_w^i, T_h^i se računaju na sljedeći način:

$$T_x^i = (G_x^i - A_x^i)/A_w^i \tag{1.5}$$

$$T_y^i = (G_y^i - A_y^i)/A_h^i \tag{1.6}$$

$$T_w^i = \ln\left(\frac{G_w^i}{A_w^i}\right) \tag{1.7}$$

$$T_h^i = \ln\left(\frac{G_h^i}{A_h^i}\right) \tag{1.8}$$

Parametri koji su označeni s G predstavljaju vrijednosti okvira oznake, dok parametri označeni s A predstavljaju vrijednosti sparenog sidrišnog okvira. Laički rečeno, T_x^i i T_y^i su omjeri diferencijala označenog i predviđenog odmaka od centra sidrišnog okvira i duljine ili visine sidrišnog okvira, te su stoga vrijednosti u rasponu od [-0.5, 0.5], dok su T_w^i i T_h^i prirodni logaritmi omjera širina i visina okvira oznake i širina i visina sidrišnih okvira, te su, ako nije došlo do pogreške u računanju G_w^i i G_h^i u rasponu od $\langle -\infty, +\infty \rangle$, no zbog prirode ovih mjera, uobičajeno je da su brojevi u neposrednoj okolini 0. Indeksi i označavaju indeks para okvira oznake i sidrišnog okvira, s obzirom na to da se računanje pronalaznog gubitka svodi na računanje glatkog L1 gubitka za svaki par.

Zaključivanje

Iz svih slojeva piramidalne mape značajki, naša mreža imat će $\sum_{l=3}^{7} W_l H_l A$ sidrišnih okvira. Razredbena mreža će za svaki okvir donijeti odluku o K brojeva, 1 za svaki razred, dok će pronalazna mreža svakom sidrišnom okviru pridodati 4 vrijednosti koje će opisivati pomak okvira graničnika od sidrišnog okvira. Radi performansi, RetinaNet uzima u obzir najviše 1000 sidrišnih okvira s najvećim vjerojatnostima za sve razrede za svaki sloj mreže piramide značajki, ali tek nakon što su se maknuli sidrišni okvir za koje je sigurnost manja od 5%.

Nakon odabira sidrišnih okvira, primjenjuje se potiskivanje nemaksimuma (engl. *non-maximum-supression*), svakom razredu posebno tako da se odabere sidrišni okvir najveće sigurnosti te se iz skupa sidrišnih okvira uklone svi sidrišni okviri kojima je presjek po uniji veći od 50%.

Nakon što su se filtrirali svi redundantni sidrišni okviri, mreža zaključuje od pomaku u odnosu na sidrišni okvir preko kojeg se može doći do vrijednosti koje bi predstavljale okvir graničnik za uočeni objekt.

Postupak pripreme i interpretacije podataka

Podaci koje predajemo mreži u svrhu učenja i zaključivanja su se prethodno morali transformirati. Baza podataka za učenje sastojala se od dvije komponente – ortopantomograma te XML (engl. *eXtensible Markup Language*) datoteka [14], po jedna za svaki ortopantomogram. Same slike bile su označavane u alatu Label Img [15], a pohranjivale su se u PASCAL VOC formatu [16]. Kako je implementacija RetinaNeta koju smo koristili preporučila korištenje CSV generatora, klase koja iz CSV [17] (engl. *Comma-Separated Values*) datoteka stvara ulazni tok podataka, podatke smo pretvorili u taj oblik.

Ovaj postupak je vrlo jednostavan – sastojao se od prolaska kroz svaku XML datoteku, knjižnicom ElementTree smo isparsirali podatke te ih pretvorili u željeni oblik, tj. metoda koja je ovo obavljala je vraćala niz znakova koji su odgovarali preslikanim podacima u CSV formatu. Potrebno je bilo sačiniti dvije CSV datoteke; prva je sadržavala tablicu s 2 stupca, u prvom je bio niz znakova kojim smo nazvali klasu, a drugi je bio cijeli broj koji je interno, u mreži, označavao klasu. Tijekom pokretanja učenja, implementacija je provjerila sve ulaze u drugoj CSV datoteci ne bi li slučajno pronašla neki razred koji se ne nalazi u prvoj datoteci. Od sad nadalje, prvu datoteku ćemo zvati *classes.csv*, a drugu datoteku *data.csv*.

Datoteka data.csv u sebi je za svaki detektirani objekt trebala sadržavati jednu šestorku; put do slike, koordinate lijevog donjeg ugla okvira graničnika (2 varijable), koordinate gornjeg desnog ugla okvira graničnika (također 2 varijable), te internu oznaku klase (drugi stupac datoteke *classes.csv* za odgovarajuću klasu). Kako je svaka naša slika imala više detekcija, tako smo morali za jednu sliku navesti i do 32 redaka informacija. Ovo je rezultiralo time da je za skupove za učenje normalnih veličina (u našem kontekstu to je bilo više od 10000 slika), *classes.csv* imao je polumilijunski broj redaka, što je jedna od neugodnosti – učenje je zahtijevalo oko 15-tak minuta od pokretanja skripte do alociranja memorije na grafičku karticu. Bilo kako bilo, nakon što smo svaki relevantan XML pretvorili u jedinstvenu CSV datoteku, bili smo spremni za učenje.

Vrijedi spomenuti da smo osim prije navedenog postupka u kasnijim testovima izveli i još par neobaveznih koraka. Inicijalno su se sve slike koje su bile označene premještale u posebnu mapu, koja bi tada poslužila kao prefiks za put do slike u data.csv. Kasnije je to riješeno tako da se u Pythonu inicijalizira klasa koja je povezivala XML datoteku sa slike i s kojom je olakšana manipulacija. Osim olakšanog pristupa, ukinuta je potreba za premještanjem datoteka. Netko bi se mogao zapitati, zašto je ovo relevantno? Tijekom premještanja, uočeno je da se nekoliko slika kopiranjem koruptiralo, međutim, implementacija RetinaNeta nije javljala nikakvu grešku, pa se sumnja da je koristila moguće kompromitirane podatke i time barem malo utjecala na učenje. O nuspojavama ćemo pričati kasnije jer bi bilo poželjno da sve kolege vide i neke od problema tijekom rješavanja ovog problema, vjerujem da će se uz naše padove netko spasiti od istih i uštedjeti vremena.

Osim pretvorbe podataka, u drugoj seriji učenja pokušali smo povećati bazu podataka za učenje i smanjiti moć šuma učenjem na modificiranom skupu podataka za učenje. Svakoj označenoj slici smo pridružili klonove kojima smo promijenili svjetlinu za -50%, -25%, +25% i +50%, te smo također stvorili klon uobičajene svjetline ali uveli Gaussov šum [18] polumjera 3. Polumjer Gaussovog šuma određen je empirijski, tj. ne postoji znanstveno opravdan razlog za biranje baš te vrijednosti izuzev „izgledalo je kao da bi takvo nešto bilo korisno“. Razlog za promjene svjetline su bile tu zbog budućih ulaza koji možda neće biti normalizirani kao skup za treniranje, dok je razlog za uvođenje Gaussovog šuma (engl. *Gaussian noise*) bila nada da će uz takvo što mreža bolje uočavati primjere koji su bili teži zbog zamućenijeg ortopantomograma. Mogli smo uvesti još modifikacija, no smatrali smo da će ovo biti dovoljno za početak s obzirom na to da nismo imali dokaza da će se tim postupcima točnost klasifikatora poboljšati.

U skupu za treniranje ostavili smo 2700 označenih slika, dok smo 300 uzeli u skup za provjeru. Kad smo obavili spomenute modifikacije, broj slika se ušesterostručio, tako da smo sada došli na 16200 označenih slika u skupu za učenje te 1800 označenih slika u skupu za provjeru.

Zaključak se vrši bez znatnih modifikacija – jednom istreniranu mrežu pohranili smo u HDF [19] datoteku (engl. *Hierarchical Data Format*) koja se naknadno

mogla pročitati te potom težine sadržane u njoj učitati u model mreže. Zatim se pokretala metoda *predict_on_batch* koja je za danu sliku vratila trojku; okvire graničnike, sigurnost zaključka te labelu. Iako bi RetinaNet trebala jednako dobro uočavati objekte različitih veličina, susreli smo se s problemom viška detekcija. Naime, kako su zubi relativno razmaknuti, dovoljno da presjeci njihovih okvira graničnika za istu detekciju ne bude veći od 50%, potiskivanje nemaksimuma koje je ugrađeno u klasifikator nije dobro obavilo posao te su teži primjeri imali više uočenih zubiju oko pravog. Ovom smo doskočili ručnim filtriranjem rezultata na sljedeći način:

Sve smo trojke pretvorili u objekt naziva *DetectedObject*, s kojim je bilo lakše manipulirati

Uzeli smo zaključke najviših sigurnosti i maknuli sve ostale oznake koje su imali veći presjek po uniji od 40% ako su ti zaključci bili istog razreda kao i onaj promatrani s najvećom sigurnosti.

Iz *DetectedObject* razreda vratili smo rezultate u trojku okvira graničnika, sigurnosti i labela. Konkretna implementacija ovog razreda nije bitna, ali važno je spomenuti da je bilo potrebno objektivizirati podatke radi lakše manipulacije.

Također je bilo razmatrano i čuvanje samo najvećih sigurnosti za svaki razred, uz iznimku implantata koji su mogli sačuvati više detekcija, s obzirom na to da možemo imati više implantata na jednom ortopantomogramu, međutim testiranjem se utvrdilo da je ovo pogoršavalo točnost klasifikatora jer je micalo neke detekcije koje su bile samo pogrešno razredbene, ali je na tom mjestu ipak bio zub. U tom slučaju se osim razredbenog gubitka računao i pronalazni gubitak, a i nama kao ljudima je bolje da je zub označen, čak i krivo, nego da uopće nije označen. Smatramo da je ovo problem koji nije rješiv filtracijom rezultata već bi trebalo implementirati rješenje koje na osnovu svih prvih zaključaka, dakle prije internog filtriranja, odlučuju koje okvire graničnike zadržati. Iako kod računalnog vida nije cilj biti ovisan o položaju predmeta na slici, u ovom slučaju bi bilo poželjno da i položaj predmeta na slici bude jedna od metrika za određivanje zuba. No to nije tema ovog rada.

Postoji još jedna specifična operacija koja je bila potrebna da bi cijeli sustav funkcionirao. Prilikom prvih zaključivanja, rezultati zaključivanja su bili vrlo loši; slika je bila sasvim krivo označena, a ponekad se, ironično, označavao i žig na ortopantomogramu. To je stvorilo neugodnu atmosferu da je učenje bilo neuspjelo, no jednim neobičnijim uzorkom utvrdili smo u čemu je riječ. Naime, kao što piše u zahvali, zubna slika kolegice bila je različite rezolucije od slika u skupu za učenje (koje se bile reda 3256 x 1536, s manjim odstupanjima, dok je slika kolegice bila razlučivosti 1600 x 853). Tada je još skup za provjeru bio veličine 104, pa samim time što je 104 slike propalo, a jedna zalutala uspjela, dovela nas je do rješenja koje smo kasnije pokušali objasniti.

Naime, slike je prije zaključivanja potrebno smanjiti. Možda je smiješno da smanjenje od svega par piksela će učiniti sustav funkcionalnim, ali to je stvarnost. Kada smo smanjili slike iz skupa za provjeru inicijalno za 1%, a kasnije i prepolovili radi performansi, stvari su radile (bolje).

Objašnjenje za ovo, koje je nadamo se ono ispravno, je princip rada mreže piramide značajki. Sjetimo se, mreža piramide značajki će smanjiti rezoluciju slike nekoliko puta. Najveća rezolucija slike koja je utjecala na mapu značajki bila je 407 x 192, dok je najmanja bila dimenzija svega 25 x 12. Vjerujemo da je ta enormna razlika u veličina rezultirala time da generirane mape značajki nisu pogodne za slike velike rezolucije. Iako je postotak koji predstavlja razliku u veličini konstantan, broj piksela je znatno različiti. Smanjenjem slike klasifikator dobiva podatke koji su sličniji onima na kojima je trenirao. Ipak smo utvrdili da postoje neke mušice i da ni ovaj sustav nije savršen. No, jednom kad se slike smanje, moguće je prilično dobro razrediti predmete na slikama, a slike su ionako inicijalno bile prevelike veličine, tj. bilo je moguće obaviti poduzorkovanje bez gotovo ikakvog gubitka informacije. Jedan od ciljeva ovog rada bilo je i istražiti koliko je moguće poduzorkovati sliku, a sačuvati točnost klasifikatora.

Slike smanjene na 1628 x 768 su na grafičkoj kartici nVidia GTX 1060 6 GB bile klasificirane u svega 500 ms. Drugim riječima, na nekoj grafičkoj kartici primjerenijoj dubokom učenju, uz podešavanje rezolucije ulaza moguće je postići klasifikaciju u realnom vremenu.

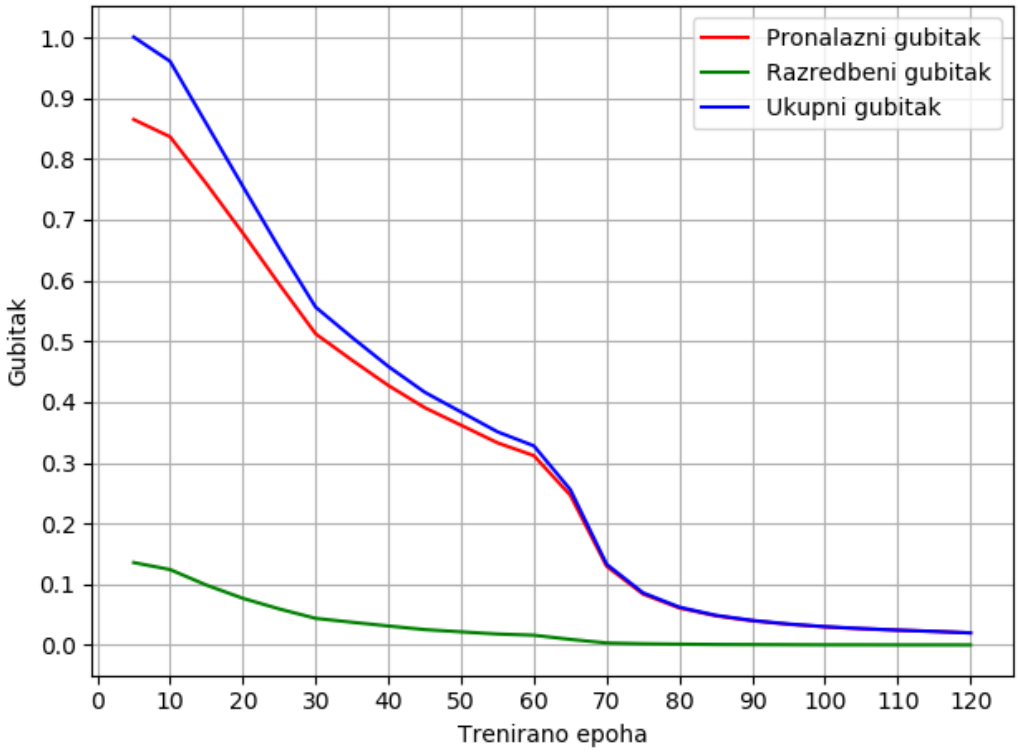
Rezultati

Za usporedbu, trenirali smo dvije mreže, odvojeno.

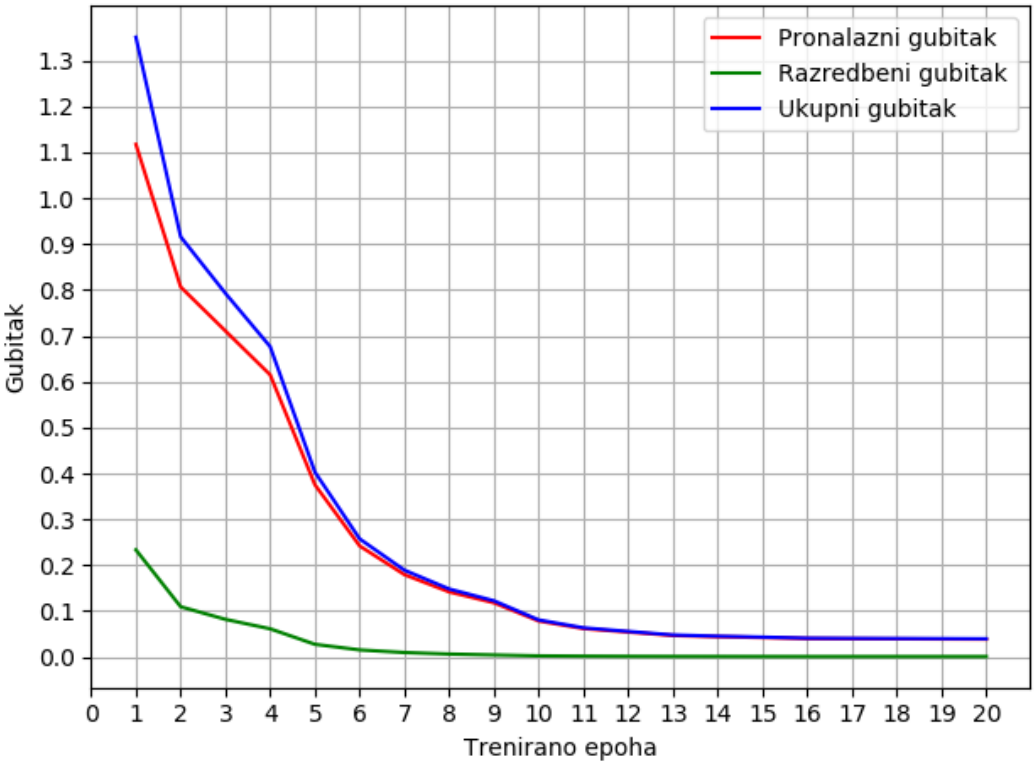
Prva mreža trenirana je skupu od 2700 slika, sve su bile nemodificirane. U tih 2700 slika bilo je i teških primjera, od primjera gdje je falio koji zub, do primjera gdje je falilo i do 10 zubi i/ili bilo zubi u iznimno lošem stanju. Ova mreža trenirana je i do 120 epoha kroz nekoliko dana, a testiranjem je utvrđeno da najbolju točnost ima snimak mreže na 85. epohi. Razlog ovome je preprilagođenost zbog malene baze podataka. Iako se 2700 slika čini kao velik broj, uzmimo u obzir da uočavamo 33 razreda predmeta na slici koja je ima ulaznu dimenzionalnost $d \cong 150000$. Kad bi pomnožili ova dva broja došli bi do 4950000 slike, no neki znanstvenici čak smatraju da bi baza podataka za učenje trebala imati barem još 5 puta više slika. Uzmemo li u obzir da bi nam za optimalno učenje tada trebalo 24750000 označenih slika, a mi imamo nešto više od 0.01% toga, ne bi trebali biti nezadovoljni rezultatima.

Druga mreža trenirana je na skupu od 16200 slika, i mreža je bila trenirana do 23 epohe. Zanimljivo je da je ova mreža puno brže konvergirala od prve, čak i ako pomnožimo broj epoha s 6. Isto kao i kod prve mreže, uzeli smo 18. epohu jer je ta imala najbolju točnost, dok su ostale iteracije nažalost bile preprilagođene.

Osim biranja mreža, ručno smo podešavali i stope učenja (engl. *learning rate*) tijekom treniranja kako bi mreža brže konvergirala prema optimalnom rješenju. Te promjene bit će označene na grafovima koji slijede u nastavku.



Slika 6: Graf ovisnosti gubitka o epohama za prvu mrežu



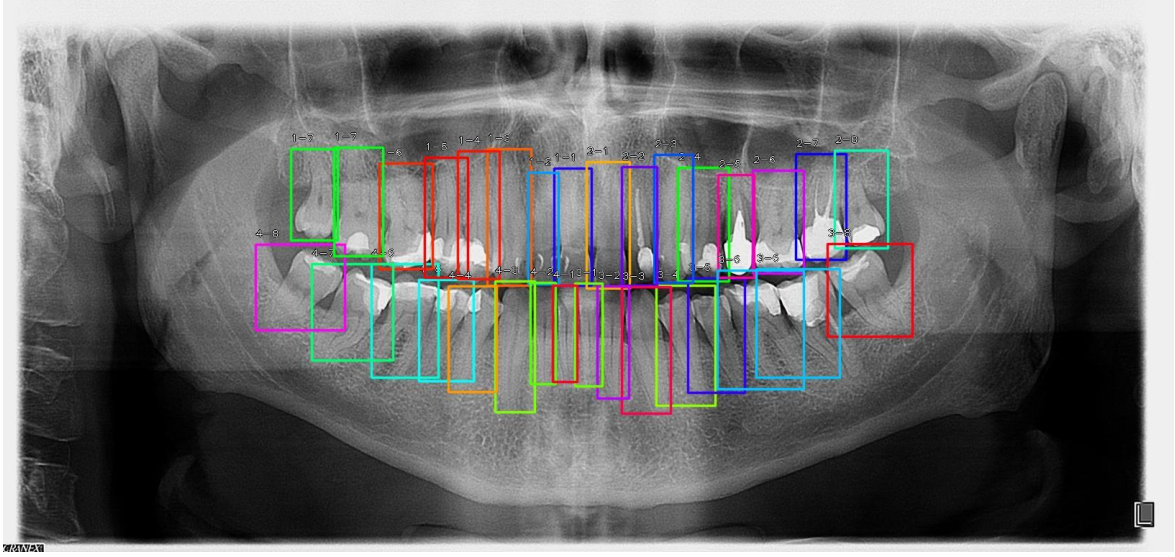
Slika 7: Graf ovisnosti gubitka o epohama za 2. mrežu

Dobiveni grafovi nastali su iz ručno spremljenih podataka; naime, uz implementaciju RetinaNeta koju imamo i korištenje CSV generatora, dodatak TensorBoard je bio onemogućen te stoga nismo mogli pratiti učenje mreže glatko kao što bi htjeli. Ipak, ovo je isključivo problem estetske prirode.

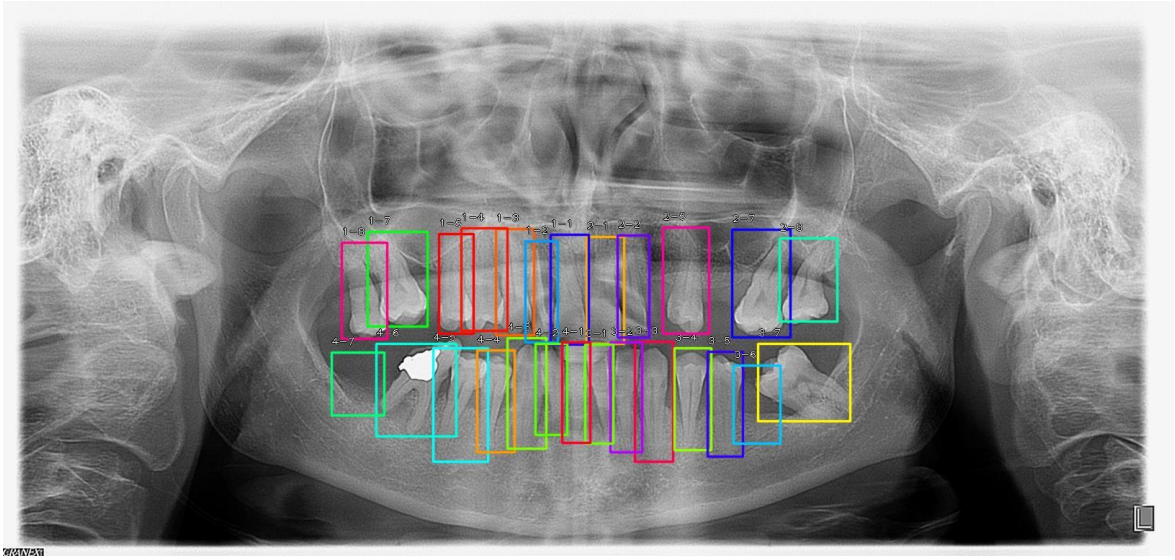
Stope učenja su povremeno ručno podešavane – sve postavke bile su uobičajene, a učenje smo započinjali stopom učenja od $3 \cdot 10^{-4}$, te je postupno smanjivali sve do 10^{-7} tijekom zadnjih epoha učenja. Stopu učenja smo bili prisiljeni mijenjati kad bi se validacijski gubitak zamrznuo na mjestu.

Na grafovima jasno možemo vidjeti da što više treniramo mrežu, ukupni gubitak postaje pronalazni gubitak. Ovo i ima smisla jer što više treniramo mrežu, ona će za razredbu nekog objekta biti sigurnija. Što je mreža sigurnija, to će razredbeni gubitak biti manji, zbog žarišnog gubitka. Nadalje, ako je nešto već krivo uočeno, šanse su da je mreža promašila i lokaciju – podsjetimo, pronalazak slijedi nakon razredbe, tj. razredba utječe na pronalazak.

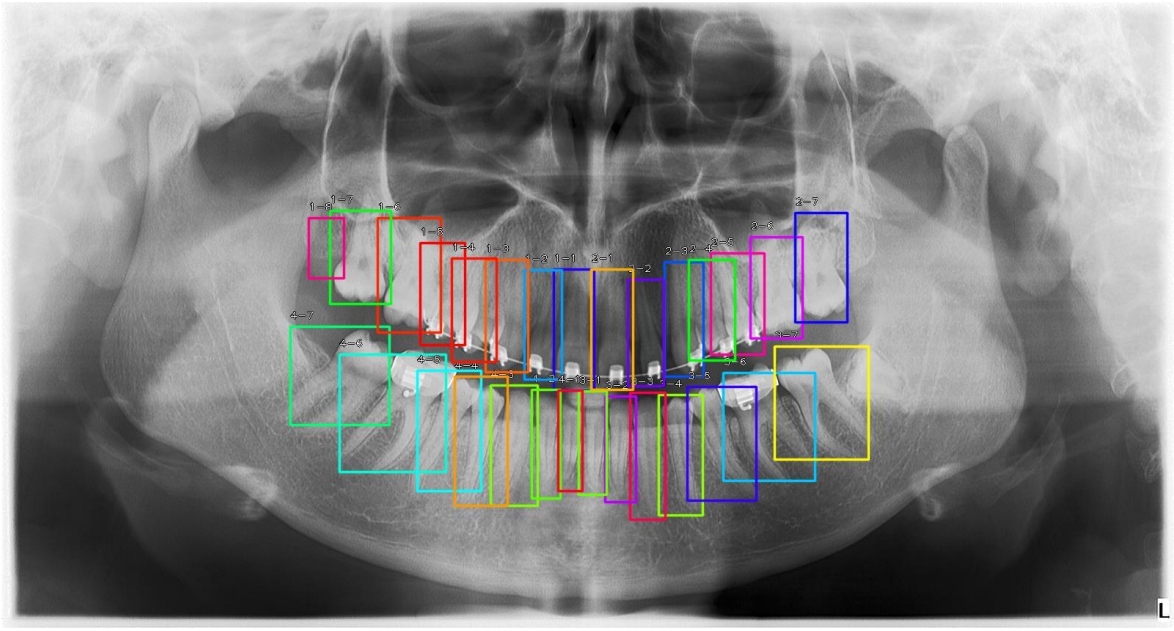
Pogledajmo malo rad ovih klasifikatora:



Slika 7: 1. ortopantomogram označen prvom mrežom



Slika 8: 2. ortopantomogram označen prvom mrežom



Slika 9: 3. ortopantomogram označen prvom mrežom

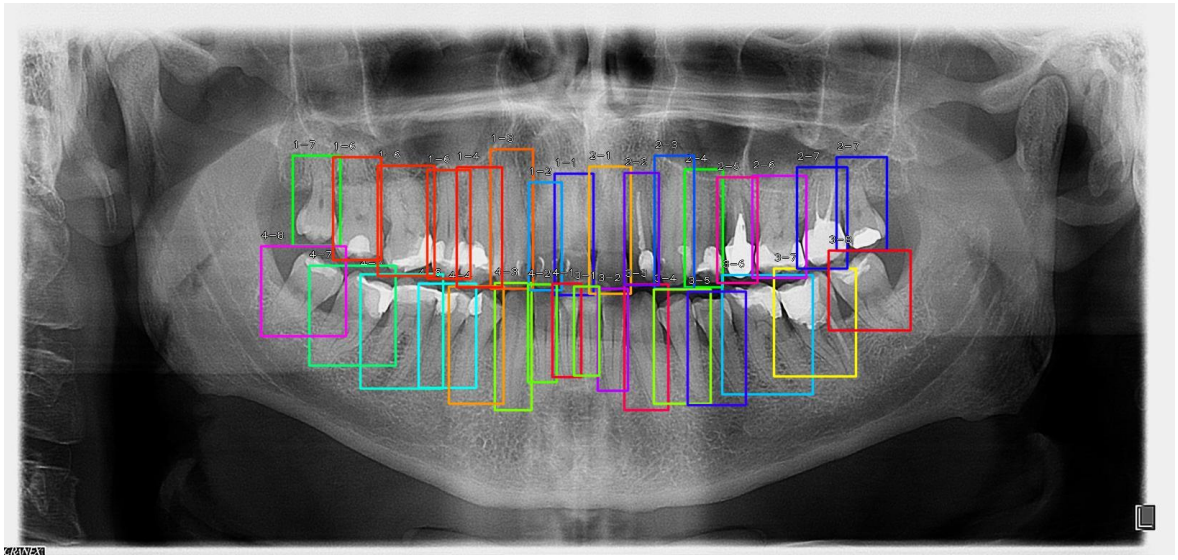
Iako se 1. ortopantomogram čini sasvim u redu, već na 2. možemo primijetiti neke čudne stvari. Na primjer, na 2. ortopantomogramu označen je zub 4-7 koji ne postoji! Ovo nazivamo problem fantomskog zuba i postoji nekoliko teorija zašto se ovo događa. Naša inicijalna sumnja bila je da mreža preprilagodi parametre za

prepoznavanje objekata na određenim lokacijama na mreži, međutim, proučavajući arhitekturu i implementaciju RetinaNeta, ona ne bi trebala imati mogućnost takvog učenja. Zatim je postojala sumnja da nešto nije u redu sa skupom za učenje; tijekom pripremanja podataka za učenje, uočene su neke neregularnosti u podacima, kao što su okviri graničnici površine 0 (1 ili više dimenzija je 0), krivo imenovane labele, labele izvan okvira. Iako se nismo pozabavili ispravnosti podataka, znamo da su neki neispravni te smo se potrudili eliminirati takve bez da ignoriramo cijelu sliku. Možda je u tome problem; eliminacijom specifičnih oznaka riskiramo zbunjivanje mreže. No, i ovo smo provjerili s par testova i došli do zaključka da je razlika zanemariva te se može pripisati nejednoznačnosti učenja, konvergencija je bila gotovo ista. Kod problematičnih slika, ako pobliže pogledamo sliku zamijetit ćemo da se ponekad može naći iznimno tanka linija, duljine 1 piksel na mjestima gdje je označen fantomski zub. Štoviše, najviše fantomskih zubi označeno je između 2 zdrava zuba, na mjestu gdje bi trebao biti zub koji je označen, ali nije jer je ili ispao, ili nije još izrastao.

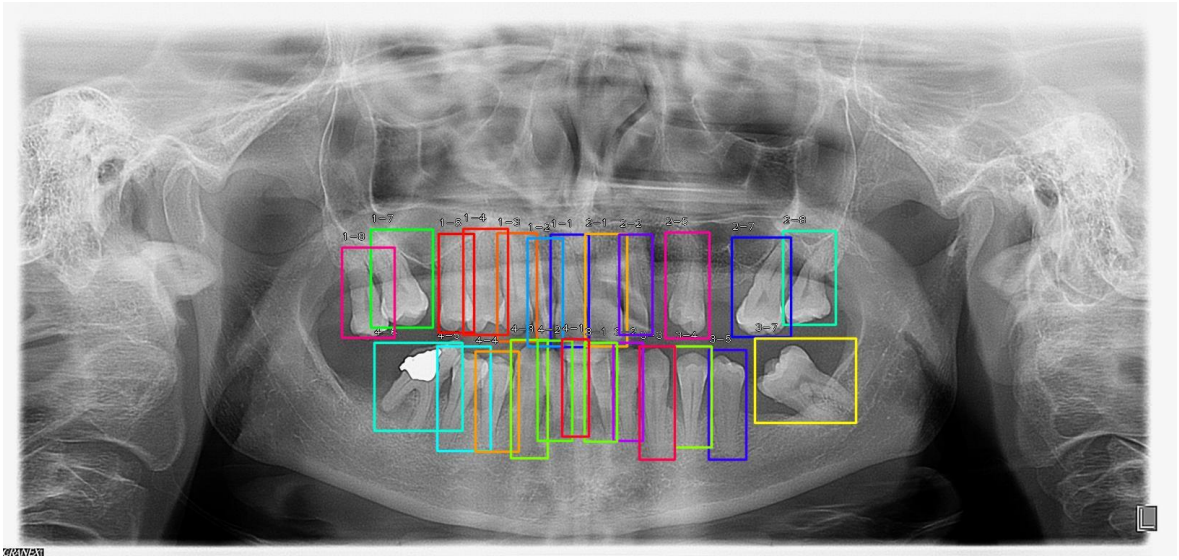
Stoga zaključujemo da bi bolje rezultate mogli dobiti provjerom i ispravkom skupa za učenje. U prethodnom radu, kada smo na ovome radili na kolegiju projekt, ovakvih problema, usprkos istoj mreži i vrlo sličnog skupu za treniranje nije bilo.

Na 3. ortopanogramu možemo zamijetiti da osim fantomskog zuba imamo i neoznačeni 3-8. Ovo nije općeniti problem, tj. pojavljuje se samo ponekad, a u ovom slučaju razlog je najvjerojatnije nekarakteristična svjetlina zuba. U nekim drugim primjerima kao potencijalni problem je otkrivena rotacija zuba. Naime, u skupu za treniranje su neizrasle osmice (1-8, 2-8, 3-8 i 4-8) uobičajeno u vodoravnoj orijentaciji. Iako rotacija nekog objekta ne bi trebala utjecati na uočavanje istog zbog prirode neuronskih, ali i konvolucijskih neuronskih mreža, moguće je da izvire problem neuravnoteženosti klasa, ali interno; klasa osmica je prezastupljena vodoravnim zubima. To bi također mogao biti i problem u 3. ortopantomogramu, ali u nastavku ćemo vidjeti zašto nije.

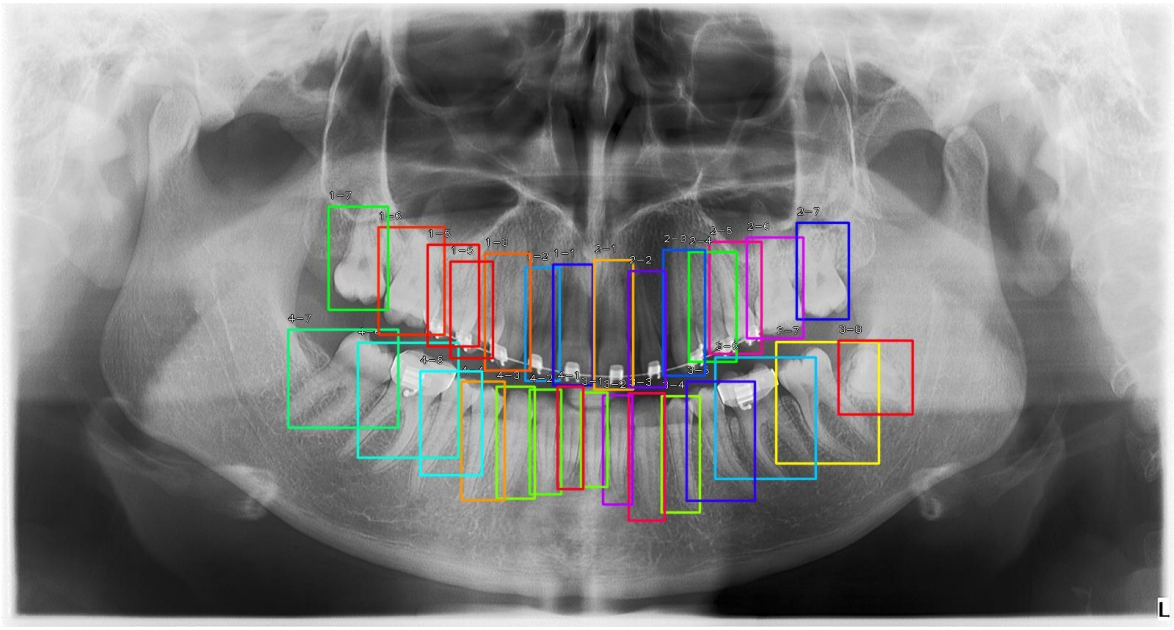
Pogledajmo rezultate za iste slike koristeći drugu mrežu, onu koju smo učili nad proširenim podacima:



Slika 10: 1. ortopantomogram označen drugom mrežom



Slika 11: 2. ortopantomogram označen drugom mrežom



Slika 12: 3. ortopantomogram označen drugom mrežom

Usporedimo li sad slike, vidjet ćemo da 1. ortopantomogram ima nešto striktnije opisane okvire graničnike, što bi se trebalo translatirati u manju pronalaznu pogrešku.

Dogodila se interesantna stvar: u 2. ortopantomogramu nestalo je fantomski zube te je sad slika potpuno ispravno označena! Naša teza je da je kod prve mreže problem bila kvaliteta slike, a ne nužno skup za učenje. S obzirom na to da je druga mreža trenirana na slikama koje su zamućene, moguće je da je upravo to zamućenje poništilo naučeni uzorak s tankim artefaktima. Bazni skup za učenje je ostao isti, tako da ako i ima nekog doprinosa od pogreški prouzrokovanih skupom za učenje, one ovdje nisu presudile.

Konačno, 3. ortopantomogram više nema fantomskog zuba, 3-8 je ispravno označen, a jedina greškica potkrala se pri označavanju zuba 1-4 koji je pogrešno označen kao 1-5. Ako dobro pogledamo sliku možemo pronaći nekoliko razloga zašto se ovo događa. Prvi razlog je aparatić. Aparatić je nešto što se može smatrati kao šum jer se ponekad pojavljuje, ponekad ne, slika je bitno različitija no razređuje se isto. Drugi razlog je sličnost zuba. Zanemarimo li okvire graničnike koji nam „zagađuju“ sliku, nije teško primijetiti da se na 3. ortopantomogramu 1-4 i 1-5 ne razlikuju bitno. Jedini način kako bi mreža mogla razlikovati između njih ako su

vizualno slični jest pozicija, što smo već utvrdili da, zbog cjelokupne arhitekture i implementacije rješenja, nije nešto što bi se trebalo dogoditi.

Dobili smo rezultate koji se na prvi pogled čine dobri – uistinu, mreža koja je bila trenirana s proširenim podacima na prvi pogled radi bolje. No, valjda to sve pretočiti u brojeve i provjeriti je li to stvarno tako na većem broju uzoraka. Uzet ćemo 300 slika iz prvotnog skupa koji nisu pripadali skupu za učenje te ćemo na osnovu rezultata ocijeniti rad mreže.

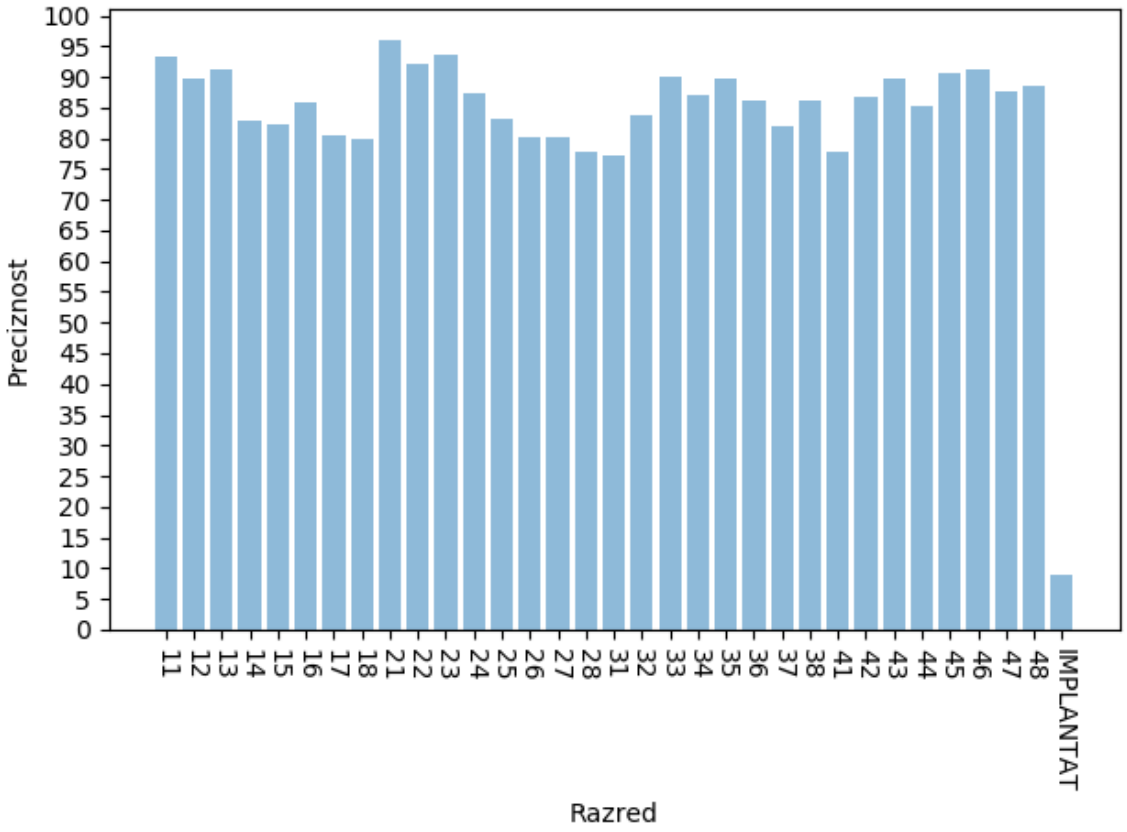
Princip ocjenjivanja točnosti mreže

Pri ocjenjivanju mreže koristit ćemo uobičajen pristup računanja prosječne preciznosti (engl. *average precision*). Postupak je sljedeći: dajemo mreži set slika na osnovu kojih će donijeti zaključke o okvirima. Usporedimo prave okvire graničnike sa zaključenima tako da im računamo presjek po uniji – za početak uzimamo prag od 50%. Sve presjeke po uniji veće od 50% računamo kao istinite pozitive (engl. *true positive*), a ostale kao lažne pozitive (engl. *false positive*). Ovo računamo zasebno za svaku klasu. Tada će prosječna preciznost svake klase glasiti:

$$AP = \frac{T_p}{T_p + F_p} \tag{2.1}$$

Gdje T_p označava istinite pozitive, a F_p označava lažne pozitive. Ako sumiramo sve prosječne preciznosti i podijelimo taj broj s brojem klasa, dobit ćemo srednju prosječnu preciznost (engl. *mean average precision*), koja nam ugrubo govori o performansama našeg klasifikatora (što viša srednja prosječna preciznost, to bolji klasifikator). Možemo postrožiti kriterije tako što ćemo povisiti prag presjeka po uniji. Uobičajeno je povisiti ga na 75%, no mi ćemo to, demonstracije radi, učiniti i za 90%.

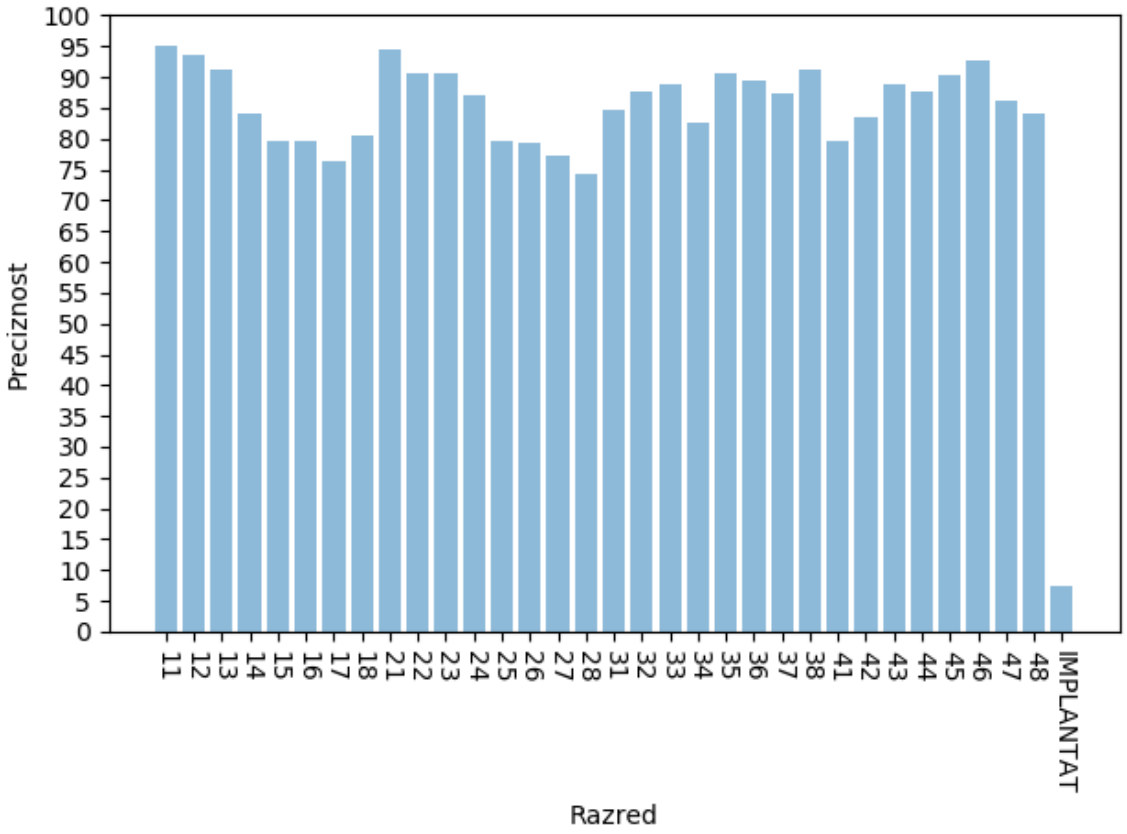
Za našu prvu mrežu, dobili smo sljedeće podatke:



Slika 13: Stupčasti dijagram prosječnih preciznosti za prvu mrežu

Zaključno, srednja prosječna preciznost za prvu mrežu je **83.8**%.

Usporedimo li ju s drugom mrežom, uz sljedeće podatke:



Slika 14: Stupčasti dijagram prosječnih preciznosti za drugu mrežu

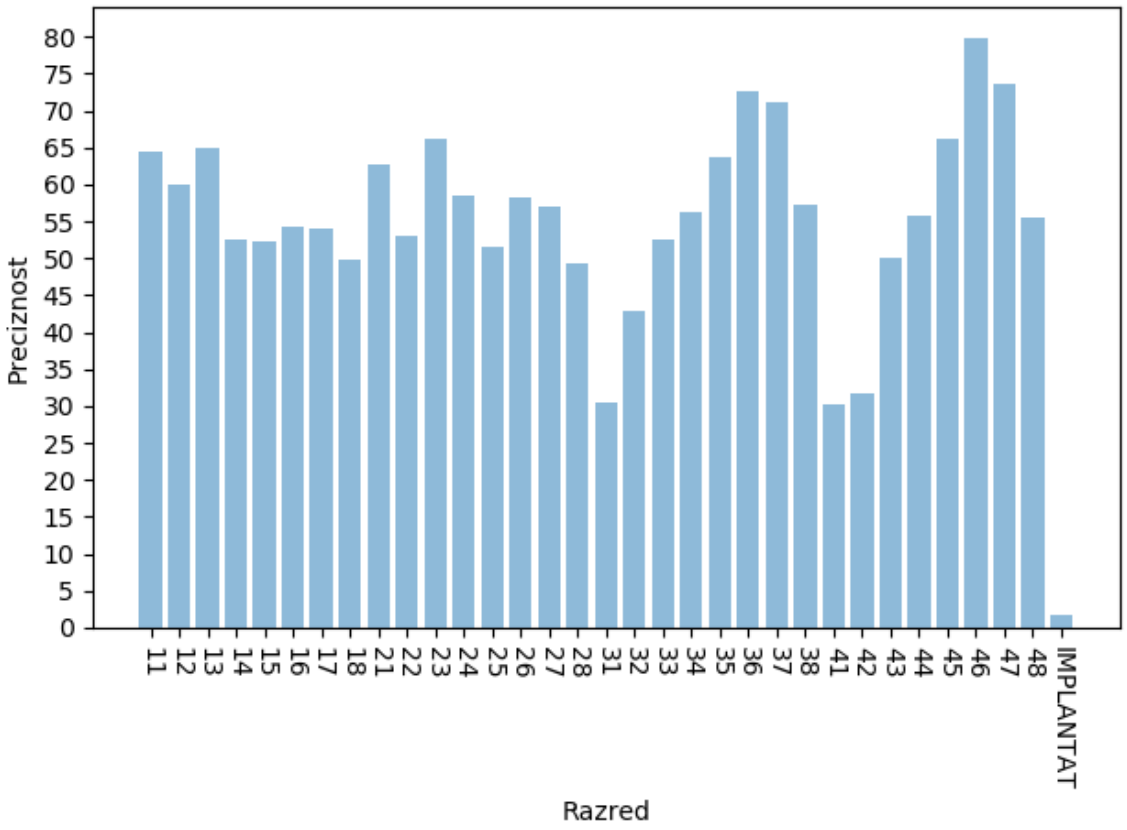
i srednju prosječnu preciznost od **83.54**%, vidimo da se ove dvije mreže u prosječnom slučaju ne razlikuju značajno. Kroz podatke je moguće vidjeti da poboljšanja u jednoj mreži rezultiraju pogoršanjima na drugim mjestima. Za jedan razred zubi jedna mreža može pobjeđivati drugu, ali će za neki drugi razred izgubiti. Ovo je vjerojatno zbog činjenice da naše modifikacije slika određene razrede zubi pogoršavaju, a druge poboljšavaju.

Valja, doduše, primijetiti i da su prosječne preciznosti za implantate iznimno loše. Ovo je zbog nekoliko faktora. Prvi faktor je to što smo neovisno o izvornom razredu zuba, sve što je bilo implantat označili i tako. Dok naš klasifikator određuje razrede zubi, svi implantati su mu jednaki, iako to s obzirom na oblik nije istina! Drugi razlog je zastupljenost implantata. Implantati su u skupu za učenje relativno rijetki. Treći razlog je nejednoznačnost pri označavanju. U skupu za učenje je puno toga označeno kao implantat. Isto tako, neki implantati su označeni kao regularni zubi.

Sve ovo je pridonijelo lošim performansama u označavanju implantata. Maknemo li loš utjecaj implantata, dobit ćemo nešto više srednje prosječne preciznosti, no to nije neka značajna razlika. S obzirom na to da se radi o jednoprolaznoj mreži koja nije bila prilagođena ovom zadatku, mislim da možemo biti zadovoljni.

Krenimo sa strožim ocjenjivanjem tako da ćemo ovog puta postaviti presjek preko unije na 75 i 90%. Ovo puta provest ćemo testiranje samo za 2. mrežu – rezultati se ne bi trebali znatno razlikovati, a i fokus ovog rada je 2. mreža.

Dobili smo sljedeće vrijednosti:

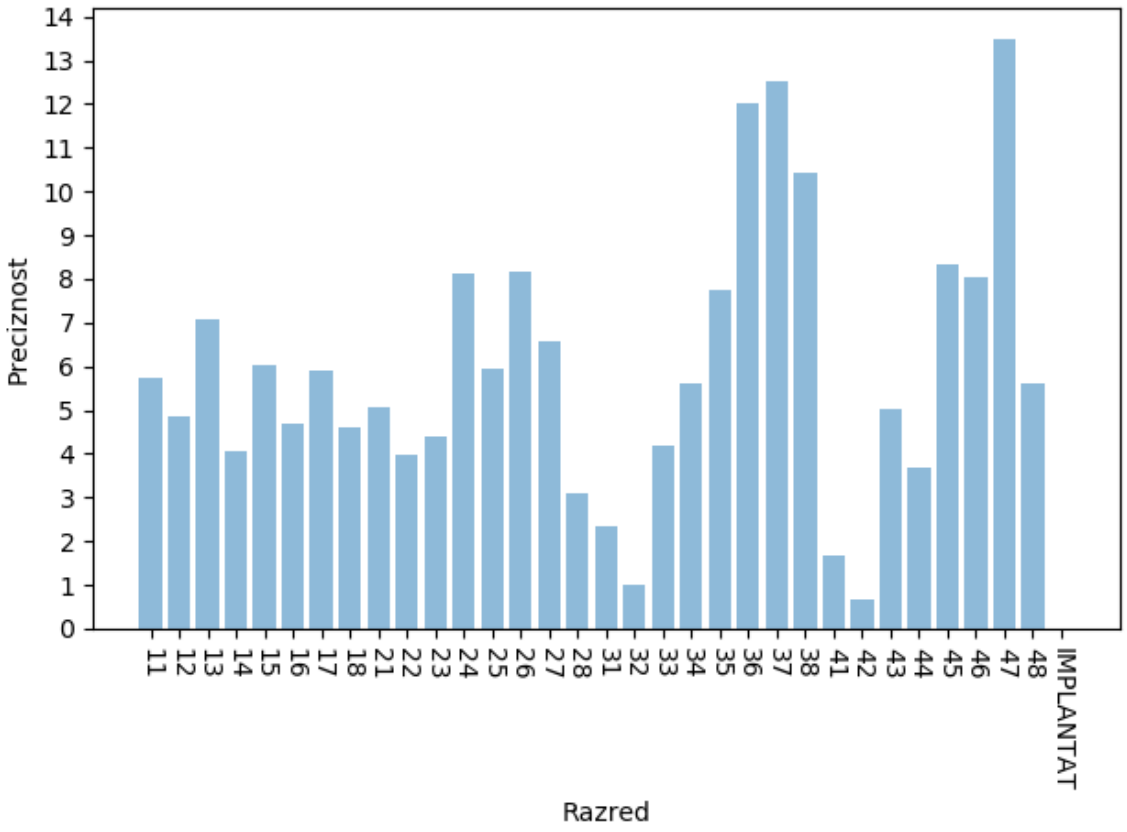


Slika 15: Stupčasti dijagram prosječnih preciznosti za drugu mrežu, IoU = 75%

Što znači da je srednja preciznost jednaka **54.55**%. Ovo je očekivano. Vidimo da neki zubi imaju iznadprosječno dobru preciznost. Ovo možemo objasniti na nekoliko načina. Možda je uobičajeno da ti zubi u skupu za treniranje imaju manje inačica, tj. da su uniformniji. Također, visoke vjerojatnosti uglavnom imaju bočni zubi. Ovo je vjerojatno zato što su veći, razdvojeniji i unikatniji, dok su zubi u sredini uobičajeno

zbijeniji i različitih oblika. Zanimljivi su, doduše padovi zubiju 3-1, 4-1 i 4-2. Radi se u sjekutićima donje čeljusti. Anegdotalno je poznato da su u skupu za učenje najteži primjeri imali iznimno zbijene sjekutiće, te je te zube ujedno bilo i iznimno teško označiti; u većini slučajeva bilo je potrebno brojati od umnjaka jer je bilo iznimno teško odrediti gdje je granica između lijeve i desne donje čeljusti, barem jednom neupućenom studentu FER-a.

Za sljedeći graf očekujemo iznimno niske vrijednosti jer se radi o pragu presjeka po uniji od 90%:



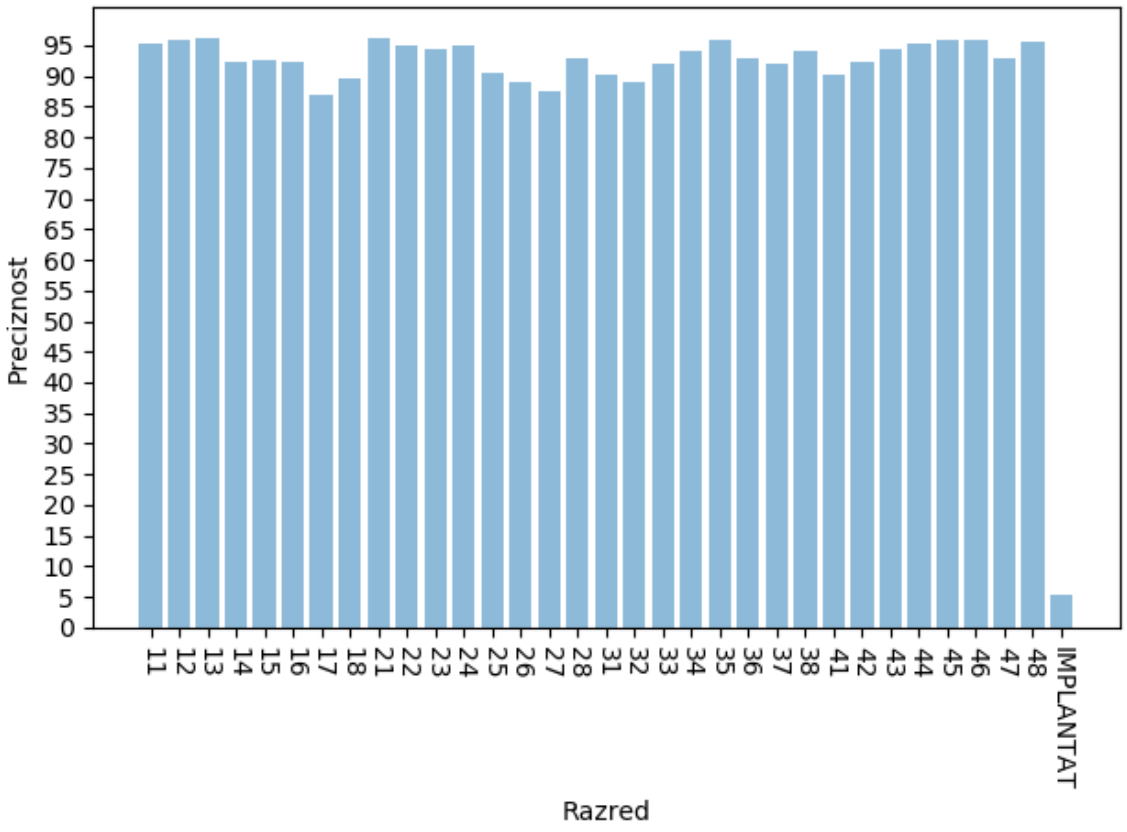
Slika 16: Stupčasti dijagram prosječnih preciznosti za drugu mrežu, IoU = 90%

Opet, specifični su donji sjekutići s iznimno niskim preciznostima. Međutim, ovoga puta jako odskaču bočni zubi donje čeljusti. Opet, naša hipoteza je da zbog veličine, razdvojenosti i unikatnosti ovih zubi klasifikator mnogo lakše određuje razrednu pripadnost ovih zubi. Ovakve niske vrijednosti ne trebaju nas obeshrabriti: na kraju krajeva, presjek po uniji od 90% je čovjeku gotovo potpun pogodak. Jedino malo

tužno je što nam je preciznost za implantate 0%: klasifikator ne uspijeva ni jednom vrlo sigurno odrediti implantat.

Određivanja praga za veličinu slike uz očuvanje točnosti

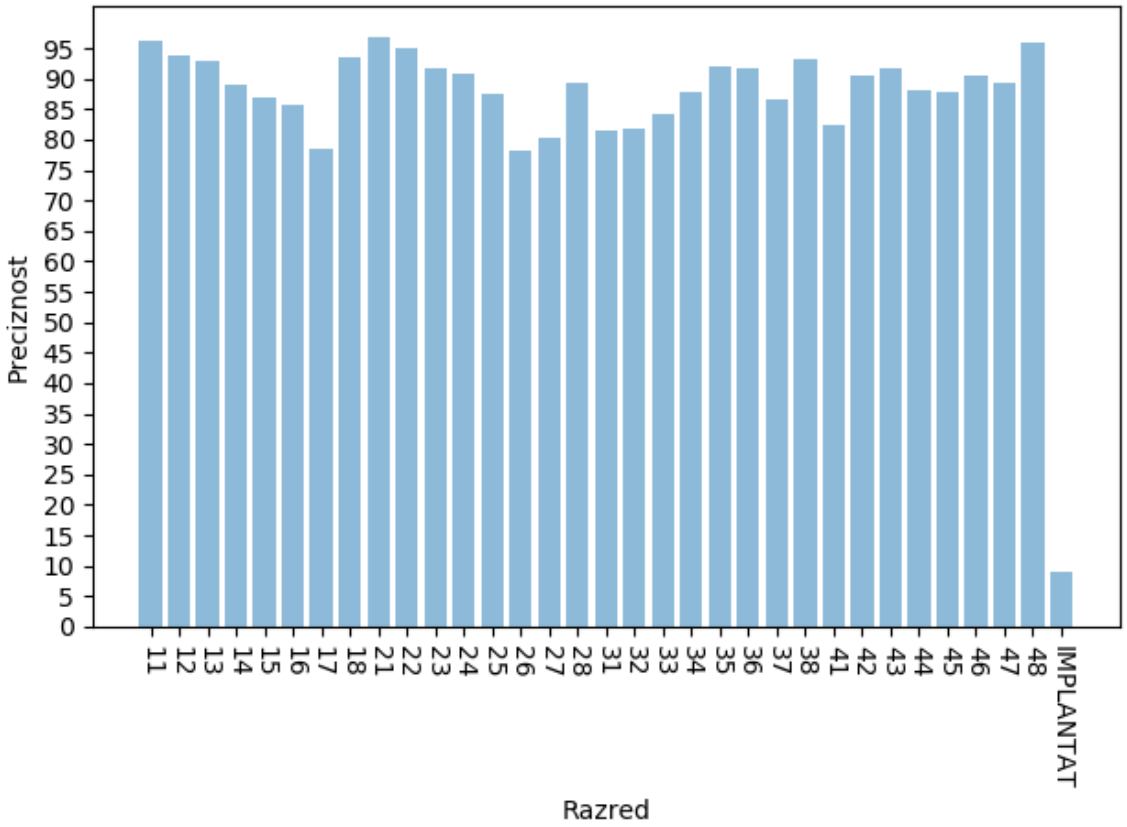
Kao posljednji dio naših rezultata pokušat ćemo smanjivati slike koliko god možemo očuvavajući prosječno preciznost. Računat ćemo preciznost za prag presjeka preko unije od 50% tako da nam prosječna preciznost ne padne ispod 80%, a pritom ćemo računati i vrijeme potrebno za obradu jedne sličice. Ponavljamo, zaključivanje se vrši na nVidijinoj GTX 1060 6 GB grafičkoj kartici, uz GPU verziju TensorFlowa, na Pythonu 3.6 i Windows 10 operacijskom sustavu. Prethodno dobiveni rezultati bili su za slike dvostruko umanjene širine i visine.



Slika 17: Stupčasti dijagram prosječnih preciznosti za drugu mrežu, slike poduzorkovane na 40% veličine

Dogodila se interesantna stvar: poduzorkovanjem slika dobili smo veću prosječnu točnost, od čak **90.26**%. Jedina stvar koja se pogoršala je preciznost za implantate, ali u vezi njih smo već gotovo odustali: takva niska preciznost je bezvrijedna. Za ovu veličinu slika, izvršavanje je trajalo 200 ms po slici, što je 5 sličica u sekundi.

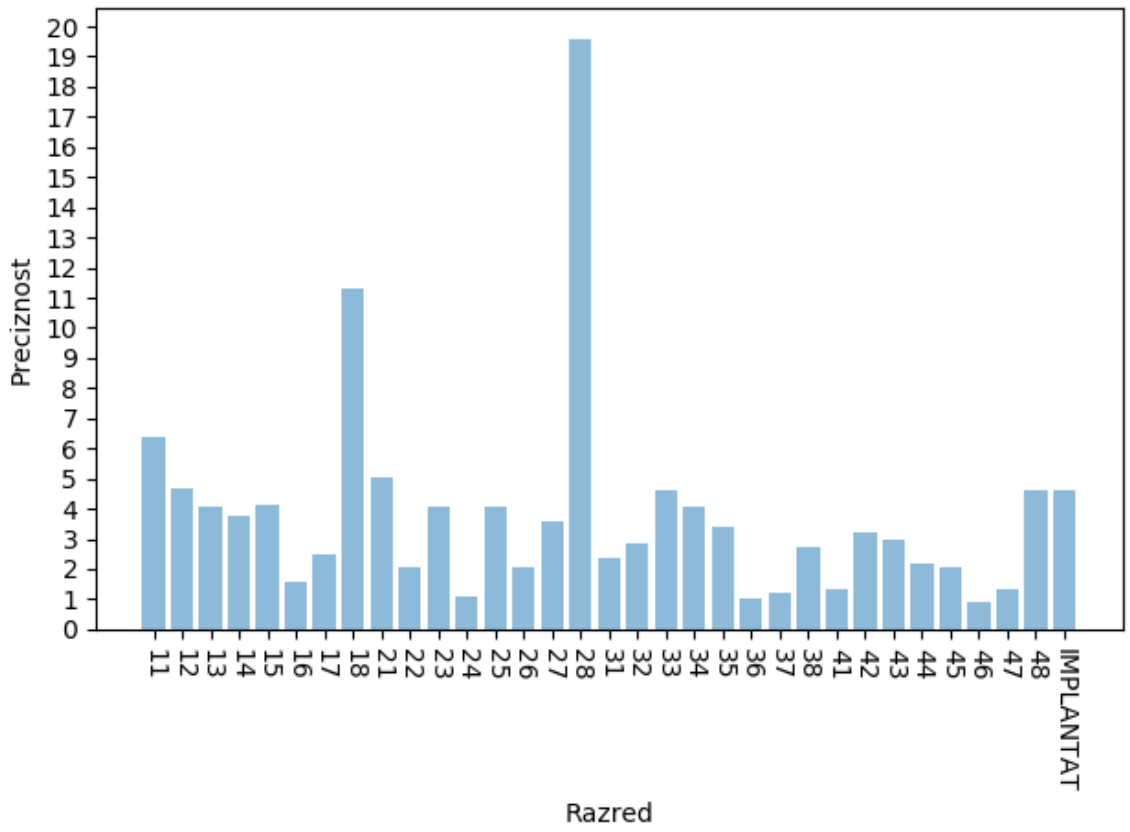
Pokušali smo smanjiti sliku na 30% veličine, ali tu preciznost već značajno opada, pa ćemo pokušati s 35% u nadi da dobijemo još bolju preciznost:



Slika 18: Stupčasti dijagram prosječnih preciznosti za drugu mrežu, slike poduzorkovane na 35% veličine

Kao što možemo vidjeti, rezultati su se pogoršali u odnosu na slike 40% veličine, sada je prosječna preciznost **86.38**%, što je još uvijek bolje od inicijalnih testova. U ovom testu izvršavanje je trajalo 130 ms po slici, što je 7.7 sličica u sekundi, što su značajno bolje performanse za relativno mali gubitak. No nažalost, tu je našem putu kraj, jer za manje slike mreža ima jako loše zaključke.

Kao posljednji test istestirat ćemo što će se dogoditi ako umjesto 50% originalne veličine zaključujemo na slikama 75% veličine:



Slika 19: Stupčasti dijagram prosječnih preciznosti za drugu mrežu, slike poduzorkovane na 40% veličine

Nažalost, manje poduzorkovanje prouzrokuje loše rezultate. Očito je da postoje neke vrijednosti koje odgovaraju mreži, s obziromna to da preciznost u ovisnosti o veličini ulaznih podataka nije linearna funkcija. Zanimljivo je kako usprkos lošim prosječnim preciznostima, u ovom slučaju je prosječna preciznost za implantatat na nivou s ostalima. Ovaj ciklus je obrađivao svaku sliku u 622 ms, odnosno 1.61 slika u sekundi. Ove performanse su za ovakvu prosječnu preciznost neprihvatljive.

Usporedba s drugim arhitekturama srodnog zadatka

Ovaj rad nije prvi svoje vrste – upravo naprotiv, ovaj problem rješava već nekoliko generacija studenata. Bilo bi zgodno uzeti ovu priliku za uspoređivanje rezultata. Primarno ćemo se referencirati na rad „Procjena zubnog statusa pomoću dubokog učenja“ [9]. Iako je inicijalno učenje obavljeno na manjem skupu za učenje, vrlo je vjerojatno da će se rezultati moći uspoređivati, kako skup za učenje nije znatno veći u smislu optimalnosti. Očekujemo da će naša mreža ipak imati značajno bolje rezultate zbog modernije arhitekture i skoro 3 puta većeg skupa za učenje.

U radu, autori su dobili prosječnu preciznost od **77.88**% za Faster R-CNN arhitekturu, **55.01**% za arhitekturu YOLO. Teoretski, RetinaNet kao jednoprolazna mreža bi se trebala uspoređivati s YOLO arhitekturom. Lagano je vidjeti da RetinaNet kao jednoprolazna mreža pobjeđuje i Faster R-CNN, dvoprolaznu mrežu. Ovo je najvjerojatnije zbog većeg skupa za učenje, no da se ovo potvrdi, bilo bi potrebno ponovno istrenirati Faster R-CNN mrežu na ovom skupu za učenje. Ono što je ključno je da usprkos tome što smo koristili novu metodologiju nismo dobili lošije rezultate – to znači da idemo u pravom smjeru. Iako nam performanse nisu ključne zbog prirode problema, mislimo da je svakom znanstveniku, novaku ili ležernom korisniku u interesu da učenje i zaključivanje bude što brže. U nastavku ćemo priložiti tablice iz prije spomenutog rada:

Tablica 1: Prosječne preciznosti za Faster R-CNN mrežu [9]

klasa	AP	klasa	AP	klasa	AP	klasa	AP
11	94,88%	21	92,44%	31	88,55%	41	84,10%
12	90,06%	22	93,04%	32	85,19%	42	85,00%
13	87,96%	23	86,04%	33	78,42%	43	85,81%
14	84,48%	24	82,93%	34	81,13%	44	75,994%
15	82,01%	25	83,71%	35	78,90%	45	82,48%
16	76,23%	26	69,61%	36	71,45%	46	68,04%
17	75,05%	27	71,40%	37	68,37%	47	72,58%
18	69,38%	28	70,83%	38	70,50%	48	71,52%

Tablica 2: Prosječne preciznosti za YOLO mrežu [9]

klasa	AP	klasa	AP	klasa	AP	klasa	AP
11	85,20%	21	24,47%	31	0,56%	41	13,82%
12	27,45%	22	33,12%	32	8,89%	42	60,48%
13	13,60%	23	18,44%	33	10,21%	43	5,99%
14	85,85%	24	75,65%	34	48,21%	44	53,48%
15	33,24%	25	53,14%	35	86,61%	45	69,68%
16	81,67%	26	77,42%	36	82,19%	46	89,14%
17	70,99%	27	69,86%	37	75,31%	47	71,08%
18	100,00%	28	96,67%	38	100,00%	48	92,89%

Zaključak

Na problemu uočavanja zuba treba još mnogo raditi. U ovom radu pričali smo o prosječnim preciznostima od 90-tak %, no što je to za jednog stomatologa? Činjenica jest da je ljudima dostupna ogromna snaga obrade. Specifično, Ahilova peta ovog problema je najvjerojatnije skup za učenje: potencijalno ogroman broj uzoraka za učenje je ono što ograničava razne algoritme i ne da im da rade gotovo savršeno. Zabrinjavajuća je činjenica da prava razredba zubiju zahtjeva još puno više razreda, tako da se potencijalno optimalna veličina skupa za učenje povećava. Ovome bi se moglo doskočiti razvojem još elegantnijih tehnika izlučivanja značajki kojima bi se dimenzionalnost ulaznih podataka mogla znatno smanjiti, a time i potreba za velikim skupom za učenje.

Što se tiče specifično arhitekture RetinaNet i našeg problema, očito je da nas arhitektura nije razočarala. Optimalne postavke, čini se, su poduzorkovanje slike na 40% veličine, ali također je očito da se mnogo toga može napraviti manipulacijama skupa za učenje – naši pokušaji proširenja skupa za učenje možda nisu urodili plodom, no očito je da se na neki način skup za učenje treba povećati.

Srećom, za ovaj problem nije potrebna prevelika snaga. Ovo nije problem nalik igranja šaha, goa, DotA-e (engl. *Defense of the Ancients*) [20][21][22] koji zahtjeva tisuće grafičkih kartica za učenje i izvođenje, ovo je relativno elegantan ali nestašan problem koji zahtjeva ponešto videomemorije i efikasnu tehniku treniranja. Budućim generacijama bismo preporučili da se više fokusiraju na transformacije skupa za učenje; tu će pronaći najviše napretka.

LITERATURA

[1] – **Artificial Neural Networks**, https://en.wikipedia.org/wiki/Artificial_neural_network

[2] – **What is a Hidden Layer?,** https://www.techopedia.com/definition/33264/hidden-layer-neural-networks

[3] – **Ciresan, Dan; Meier, U., Schmidhuber, J**., Multi-column deep neural networks for image classification, 2012 IEEE Conference of Computer Vision and Pattern Recognition, (2012.), str. 3642-3649.

[4] – **Adit Deshpande**, A Beginner's Guide To Understanding Convolutional Neural Networks, https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[5] – **Karpathy, A., Johnson, J.**, Standfordov CS231N tečaj, http://cs231n.stanford.edu/

[6] – **Activation function**, https://en.wikipedia.org/wiki/Activation_function

[7] – **Loss function**, https://en.wikipedia.org/wiki/Loss_function

[8] – **Ren, S., He, K., Girshick, R. et al,** Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, https://arxiv.org/abs/1506.01497

[9] – **Ilić, I., Koledić, K., Drabić, M. et al**, Procjena zubnog statusa pomoću dubokog učenja, FER, 2018.

[10] – **Redmon, J., Divvala, S., Girshick, R. et al**, You Only Look Once: Unified, Real-Time Object Detection, https://arxiv.org/abs/1506.01497

[11] – **Lin, T-Y., Goyal, P., Girshick, R. et al**, Focal Loss for Dense Object Detection, https://arxiv.org/abs/1708.02002

[12] – **Zeng, N**., RetinaNet Explained and Demystified, https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/

[13] – **He, K., Zhang, X., Ren, S. et al**, Deep Residual Learning for Image Recognition, https://arxiv.org/abs/1512.03385

[14] – **XML**, https://en.wikipedia.org/wiki/XML

[15] – **Label Img**, https://github.com/tzutalin/labelImg

[16] – **Pascal VOC**, http://host.robots.ox.ac.uk/pascal/VOC/

[17] – **Comma-separated values**, https://en.wikipedia.org/wiki/Comma-separated_values

[18] – **Gaussian noise**, https://en.wikipedia.org/wiki/Gaussian_noise

[19] – **Hierarchical Data Format**, https://en.wikipedia.org/wiki/Hierarchical_Data_Format

[20] – **AlphaZero**, https://en.wikipedia.org/wiki/AlphaZero

[21] – **AlphaGo**, https://en.wikipedia.org/wiki/AlphaGo

[22] – **OpenAI**, https://openai.com/five/

Sažetak

Detekcija zuba u ortopantomogramima

Metodama dubokog učenja uz arhitekturu RetinaNet istražujemo jednoprolaznu neuronsku mrežu i njen učinak na problemu detekcije zuba u ortopantomogramima. Prvo se upoznajemo s najosnovnijim konceptima neuronskih mreža općenito, pa krećemo u specifičnosti našeg zadatka. Uspoređujemo različite pristupe problemu uz bazu podataka za učenje koja sadrži zube koji pripadaju širokom rasponu ljudi. Dolazimo do optimalne srednje preciznosti od **90.26**%, što je nakon usporedbe bolje od ijednog rezultata srodnih radova do sad. Tijekom svega ovoga razmatramo o stvarima koje su nas sputavale ili o pretpostavkama kada ne znamo sigurno. Pronalazimo potencijalna rješenja za neke od problema s kojima se susrećemo. Na kraju sagledavamo rješavanje ovog problema s pogleda performansi.

Ključne riječi: duboko učenje, neuronske mreže, ortopantomogrami, RetinaNet, žarišni gubitak

Summary

Teeth detection in orthopantomograms

Using deep learning methods and the RetinaNet architecture we explore a single pass neural network and it's efficiency regarding the teeth detection problem. The first step is understanding the most basic neural network concepts after which we delve into the specifics of our task. We compare different approaches to our problem using our training database which contains teeth belonging to a wide spectrum of people. We derive the optimal average precision of **90.26**%, which ends out being the best one when compared to all related scientific work in the same faculty. During all of this we discuss things that may have slowed us down or think about it when we're not exactly sure where the problem lies. We find potential solutions for some of the problems we face. At the end we look at this problem from a performance perspective.

Key words: deep learning, neural networks, orthopantomograms, RetinaNet, focal loss

_____

potpis kandidata