Usporedba metoda predobrade slika za segmentaciju srčanih komora iz 2D CT slika pomoću U-Net konvolucijske neuronske mreže

Ćaćić, Daria

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstva

USPOREDBA METODA PREDOBRADE SLIKA ZA SEGMENTACIJU SRČANIH KOMORA IZ 2D CT SLIKA POMOĆU U-NET KONVOLUCIJSKE NEURONSKE MREŽE

Diplomski rad

Daria Ćaćić

Osijek, 2023.

SADRŽAJ

1. UVOD

CT ili računalna tomografija koristi se za rekonstrukcija dijelova tijela te predstavlja jednu od važnijih metoda koja se koristi pri medicinskoj dijagnostici već godinama. Kako bi različita područja bila jasnije naglašena korisno je provesti segmentaciju na CT slikama. Segmentacija će učinit željene regije uniformnima s obzirom na neko zajedničko svojstvo te granice jednostavnima, dok se susjedne regije značajno razlikuju. U-net konvolucijska neuronska mreža kvalitetno obavlja segmentaciju medicinskih slika, ali rezultati znatno ovise ne samo o metodi treniranja modela nego i o podatkovnom setu korištenim za treniranje. Slike dobivene CT uređajem zbog šumova i samih ograničenja stroja imaju određenu kvalitetu što može uzrokovati otežano raspoznavanje različitih područja. Kako bi se riješio taj problem obavlja se predobrada podataka čime se poboljšava kvaliteta slike te olakšava treniranje nad njima. U ovome radu razmatra se nekoliko pristupa tom problemu te se korištenjem metoda predobrade slike trenira U-net model te se na kraju uspoređuju dobiveni rezultati. Svaki korak izrade detaljno je pojašnjen uz teorijsku pozadinu problema i programskog rješenja u programskom jeziku *Python*.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je istražiti i opisati način dobivanja CT slika i njihove karakteristike. Potrebno je istražiti i opisati kliničku pozadinu (prikaz srca na CT slikama, dijelovi srca, klinička potreba) te dati kratki pregled područja. Objasniti teorijske osnove načina rada konvolucijskih neuronskih mreža. Razviti sustav za predobradu CT slika. Razviti sustav za segmentaciju cijelog srca pomoću U-Net konvolucijske neuronske mreže. Usporediti dobivene rezultate segmentacije s i bez pretprocesiranja podataka. Prikazati i objasniti rezultate te odrediti preciznost izvođenja razvijenog sustava.

2. PREGLED PODRUČJA

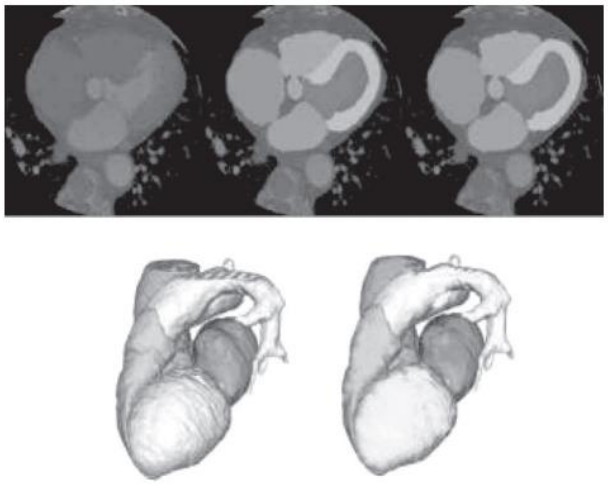
Od pojave U-net konvolucijske neuronske mreže 2015. godine koja je predložena kao jedno od rješenja za segmentaciju medicinskih slika napisano je nekoliko članaka u kojima prikazuju kako se predložena metoda koristi i kakve rezultate daje za nekoliko različitih područja snimanja (segmentiranje različitih organa) te za različite uređaje (CT i magnetska rezonanca). Hsin-Han Tsai i suradnici [2] predstavili su ideju korištenja područja od interesa (*engl. region of interest (ROI)*) kako bi se poboljšala performansa u zaključivanju te je fokus na segmentaciji jetre. Zbog otežanog treniranja modela za segmentaciju na podatcima koji sadrže cijeli volumen jednog presjeka tijela uključujući jetru ili bez nje model se trenira samo na slikama jetre te se naknadno pomoću metode računalnog vida odabere područje presjeka na koje se aplicira istrenirani model. Pokazano je da ovaj pristup može poboljšati točnost do određene razine te da je potrebno razvijati nove metode koje će moći ostvariti rezultate veće preciznosti.

Nadalje, radovi se pretežito oslanjaju i na korištenje U-net neuronske mreže za segmentaciju medicinskih slika dok je broj slobodno dostupnih radova fokusiranih na metodama predobrade jako malen. Najznačajniji je rad autora Islam i suradnici [3] gdje je naglašen problem sličnih intenziteta svjetlosti jetre i susjednih područja, razine kontrasta, šum uzrokovan medicinskim uređajima te nepravilnih oblika jetre pri korištenju slika jetre dobivenih CT uređajem i MRI uređajem. Segmentira se samo jetra u odnosu na organe koji ju okružuju, ne njeni dijelovi. Za predobradu korištene su metode HU-*windowing* CLAHE, *z-score normalization*, median filter i *Block-Matching* 3D filter (BM3D). Korištenjem zasebno i kombiniranjem različitih prethodno spomenutih metoda najbolji rezultat proizašao je kombiniranjem HU-windowing-om, popraćen median filterom i z-score normalizacijom.

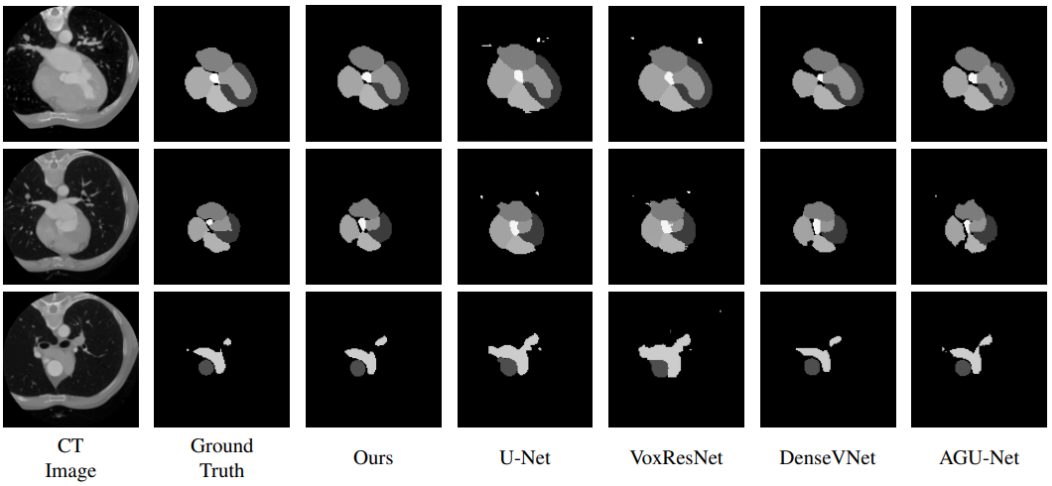
Postoji i nekoliko radova na temu segmentiranja CT slika srca. Jedan od tih je autora Habijan i suradnici [4] gdje se koristi U-net neuronska mreža za segmentaciju srčanih komora 3D slike dobivene CT uređajem bez korištenja predobrade (Sl. 2.1.). Kombinirane su dvije 3D U-net arhitekture, jedna za lokalizaciju bounding box-a oko srca i druga za segmentaciju. Rezultati ukazuju da je predložen pristup prikladan za segmentiranje srca.

Nadalje, u radu autora Yoshida i suradnici [5] procjenjuje se korisnost korištenja metoda dubokog učenja za segmentaciju dijelova srca pedijatrijskih pacijenata mlađih od 15 godina dobivenih CT uređajem korištenjem U-net mreže. Rezultati ukazuju da U-net mreža daje visoku točnost te ova metoda može biti korisna i kod korištenja na pedijatrijskim pacijentima.

U radu autora Park i Chung [6] predstavljena je problematika detaljnijeg segmentiranja srca prikazivajući dijelove kao miokard zbog bliskosti i sličnosti određenih dijelova što otežava segmentaciju (Sl. 2.2.). Predstavljen je model koji rješava taj problem korištenjem shape-aware attention metode te uspoređen s drugim contour-based attention metodama. Korištena metoda prikazala je poboljšanje u točnosti spram prijašnjih.



Sl. 2.1. Rezultat segmentacije iz rada „Neural Network Based Whole Heart Segmentation from 3D CT Images“ [4]



Sl. 2.2. Rezultat segmentacije iz rada „Cardiac Segmentation on CT Imagesthrough Shape-Aware Contour Attentions“ [6]

3. TEORIJSKA OSNOVA

3.1. CT slika i klinička pozadina

Računalna tomografija odnosi se na proceduru dobivanja slike korištenjem x zraka usmjerenih na pacijenta koje se brzo rotiraju oko tijela proizvodeći signale koji su obrađeni u računalu kako bi se dobile slike presjeka tijela. Slike dobivene na taj način mogu dati više informacija od konvencionalnih x zraka. Nakon što se prikupi određen broj uspješno dobivenih slika računalo ih rekonstruira i formira 3D sliku koja se kasnije koristi za medicinsku dijagnostiku.

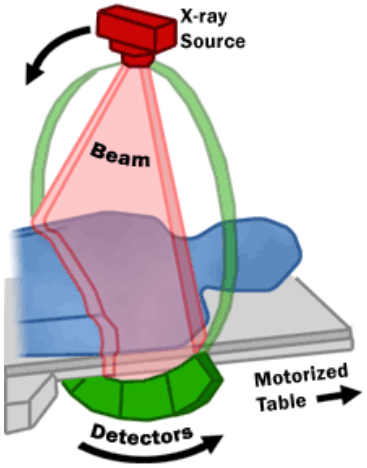
Jedna od češćih upotreba CT-a kod snimanja srca je za detekciju protoka kroz koronarne arterije. Računalna tomografija (Sl. 3.1.) može dobro prikazati funkcije, srčane krvne žile i zaliske.

Sl. 3.1. Moderan CT stroj [7]

Precizno prikazuje blokade u protoku kroz određene segmente krvnih žila. Glavni nedostatak je velika količina zračenja zbog čega se ne treba često koristiti te se najčešće koristi kao sekundarna ili tercijarna metoda medicinske dijagnostike (nakon elektrokardiograma i ultrazvuka srca) ako se pokaže potreba za time. Jedan od također potencijalno opasnih posljedica je moguća alergija na kontrastno sredstvo ili oštećenje bubrega zbog kontrastnog sredstva. CT srca se koristi nekad i da bi se proizvela 3D slika srca na kojoj se dalje dijagnosticira.

3.1.1. Načini dobivanja CT slike

Za razliku od konvencionalnih x zraka gdje se koristi fiksni izvor x zraka, CT skener koristi motoriziran rotirajući izvor (Sl. 3.2). Za vrijeme snimanja pacijent leži mirno dok se oko njega sporo rotira skener i šalje x zrake kroz tijelo. Umjesto filma koriste se detektori koji su locirani suprotno od izvora x zraka. Kako x zraka izlazi iz tijela, detektori ju prikupljaju i prenose u računalo. Svaki put kada izvor zraka obavi čitav krug, računalo spojeno na CT koristi sofisticirane matematičke izračune kako bi konstruirao dvodimenzionalnu sliku pacijenta. Debljina tkiva na svakoj slici može varirati ovisno o CT stroju koji se koristi, a obično iznosi između 1 i 10mm. Nakon što se snimi jedna slika, što obično traje oko 1s, ona se sprema te ležaj pomiče kako bi se snimio sljedeći presjek. Postupak se nastavlja dok se ne dobije odgovarajući broj presjeka. Gusta tkiva kao kost lako su vidljiva na slici dok u slučaju mekih tkiva poput tkiva organa varira ovisno o tipu. Zbog toga se u nekim slučajevima koristi i kontrast kako bi tkiva postala vidljivija. Za krvožilni sustav koristi se najčešće kontrast baziran na jodu dok se za probavni sustav koristi kontrast baziran na bariju. Kontrast blokira x zrake tako da ta mjesta budu bijela na slikama. U tablici 3.1. mogu se vidjeti CT brojevi nekih tkiva u ljudskom tijelu.



Sl. 3.2. Kretnje CT stroja [8]

3.1.2. NifTI format

Tablica 3.1. CT broj nekih tkiva u ljudskom tijelu

Element	*vrijednost*
Zrak	-1000
Mast	-50 do -100
Voda	0
Meko tkivo	30 do 80
Zgrušana krv	60 do 90
Pluća	-400 do -600
Krv	40
Jetra	40 do 60
Bijela tvar	-20 do -30
Tamna tvar	-37 do -45
Kosti	400 do 1000 i preko
Jodni kontrast	100 do 500, ovisi o koncentraciji

Već prethodno spomenuto je da format obično bude u 3D, dok se ovdje radi s obradom 2D slika. Prije obrade 2D slike potrebno ih je dobiti iz 3D formata. Format 3D slike koji je ovdje korišten je NIFTI s nastavkom .nii. NIftI je kratica za Neuroimaging Inforatics Technology Initiative koja je također i naziv udruge koja je kreirala ovaj oblik datoteke ranih 2000-tih. NIfTI

sadrži header datoteku i slikovnu datoteku. Ovaj tip datoteke najčešće se koristi u medicinskoj radiologiji zajedno sa DICOM formatom. Pretvorba iz jednog u drugi format je također moguća pomoću odgovarajućih formata. Ovaj format je nastao iz Analyze formata prethodno korištenog za neurološke slike čiji nedostatak je bio manjak informacija o orijentaciji. Te informacije su morale biti spremite u zasebnoj datoteci. Slike analyze formata se spremaju u .hdr i .img kako se može spremiti i nifti format, ali nifti se može spremiti i u jednoj datoteci .nii. Nifti format je kompatibilan sa analyze te ima isti header s tim da ne koristi sve prepisane vrijable ili ih koristi drukčije.

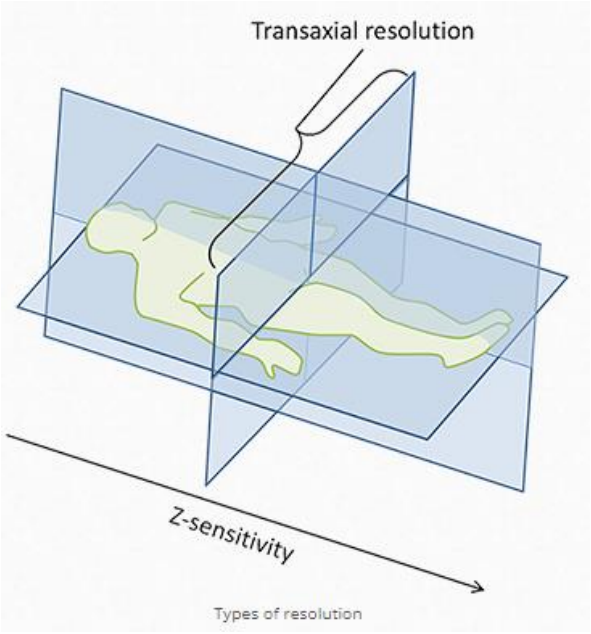
3.1.3. Karakteristike CT slike

Način rada CT uređaja temelji se na upotrebi niza x zraka koje se šalju pod različitim kutovima te se slika kreira mjerenjem razlike između emitiranih i apsorbiranih zraka kroz tijelo. Ta razlika je ateunacija čija je vrijednost određena gustoćom tkiva. Kako različita tkiva imaju različitu gustoću, kreiraju se slike koje imaju različit intenzitet boje. Što je materijal gušći to je slika svjetlija. Vrijednost skale 0 predstavlja vodu. Meka tkiva unutar tijela imaju manju gustoću te je zbog toga njihov prikaz tamniji. Kosti, koje imaju veću gustoću, prikazane su svjetlije. Svakom tkivu pridružena je vrijednost sa Hounsfieldove skale koja je univerzalna za sve CT uređaje. Taj broj opisuje rendgensku vidljivost (engl. *radiodensity*) i naziva se Hounsfieldov ili CT broj. CT slike većinom koriste 12-bitne datoteke za pohranu vrijednosti koje su između -1024 i 3071. CT slika daje tri različita presjeka ovisno o kutu snimanja. Ti presjeci su koronarni, saginalni i

aksijalni. Pošto ljudsko oko može samo ograničen broj nijansi sive percipirati, promatrač može odabrati raspon Hounsfieldovih brojeva preko kojih će se primijeniti *grayscale*. Taj raspon se zove prozor i prilagođava se tipu tkiva i različitim potrebe preciznosti.

Kvaliteta medicinskih slika je važna jer medicinska dijagnostika zahtjeva veliki stupanj točnosti. Kvaliteta CT slike određena je trima faktorima: rezolucijom, količinom šuma i kontrastom. Rezolucija je mjera koliko dva objekta trebaju biti udaljena kako bi se mogli vidjeti kao zasebni objekti na slici. Kako bi se vidjeli kao zasebni objekti mora se odrediti njihova granica. Postoje dvije rezolucije u CT slikama: transaksijalna i Z-osjetljivost.

Minimalna transaksijalna rezolucija određena je stvarnom veličinom detektora ili njegovom širinom (Sl. 3.3.). Što je manji detektor to je veća rezolucija. Na nju utječu hardverski faktori skenera i parametri rekonstrukcije. Faktori skenera su žarište koje utječe tako da što je manje to



Sl. 3.3. Prikaz dviju rezolucija CT slike [11]

je veća rezolucija, veličina detektora gdje manji detektori daju bolju rezoluciju, ali što je veća gustoća detektora u ograničenom prostoru to je veća mogućnost od mrtvog prostora te se smanjuje efikasnost. Parametri rekonstrukcije su broj projekcija, filter rekonstrukcije i veličina piksela. Kod broja projekcija veći broj daje bolju rezoluciju. Rekonstrukcijski filter utječe tako da „oštriji“ kerneli daju bolju rezoluciju te uz to proizvode dosta šuma zbog ne poništavanja velike prostorne frekvencije. Veličina piksela u milimetrima dana je formulom $d = \frac{FOV}{n}$ gdje je FOV „*field of view*“ dan u milimetrima i n veličina jezgre slike. Najveća prostorna frekvencija

(*fmax*) još zvana Nyquistov limit dana je formulom $fmax = \frac{1}{2*d}$ iz koje se vidi da veća veličina piksela daje manju maksimalnu prostornu frekvenciju. Da bi se povećala prostorna frekvencija može se smanjiti FOV što smanjuje veličinu piksela po prvoj formuli ili povećati veličinu jezgre tj. što je veći *n* to je manja veličina piksela.

Z-osjetljivost se odnosi na efektivnu širinu *slice-a* slika. Faktori koji utječu na to su detektor debljine *slice-a*, preklapanje uzoraka i žarište. Što je širi detektor po z osi to je manja rezolucija. Širina slice-a na rezoluciju utječe tako da veća širina dalje bolju rezoluciju, ali stvara više šuma dok tanji slice-vi omogućavaju izotropsko skeniranje čije su prednosti smanjeni parcijalni volumni efekt, bolje višeplanarsko reformatiranje i poboljšano 3D prezentiranje podataka.

Sljedeći faktor koji utječe na kvalitetu je šum. Šum je jedan od glavnih problema u obradi slike te ga je važno što bolje otkloniti, posebno zbog ozbiljnosti medicinske dijagnostike. Glavna tri izvora šuma su kvantni šum, električni šum i šum izazvan procesom rekonstrukcije od kojih je primarni izvor kvantni. Pri detekciji fotona doći će do varijacije u broju detektiranih fotona i taj broj će varirati nasumično oko prosječne vrijednosti te ta varijacija predstavlja šum. Taj šum čini sliku zrnastom. Povećanjem broja fotona smanjuje se šum. To se može postići povećavanjem struje kroz cijevi, povećavanjem rotacijskog vremena, povećavanjem širine slice-a te povećanjem voltaže. Sve vrijednosti osim promjene voltaže direktno su proporcionalne povećanju fotona dok povećanje voltaže nije direktno proporcionalno.

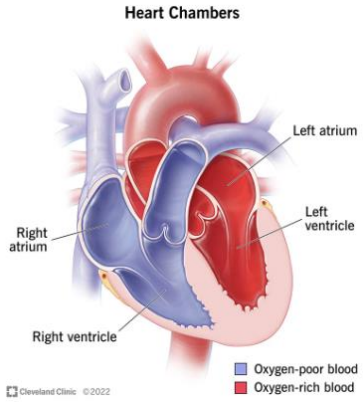
Zadnji faktor koji će se spominjati je kontrast. Na njega utječu šum, struja, inherentna svojstva tkiva, voltaža i korištenje kontrastnog sredstva. Šum utječe tako da što ga je više stvara gori kontrast dok struja što je niža utječe na stvaranje više šuma čime također pogoršava kontrast. S povećanjem šuma smanjuje se kvaliteta slike, a smanjenje struje uzrokuje povećanje šuma. I svojstva tkiva mogu utjecati svojim vrijednostima ateunacije te time učiniti otežano razaznavanje između različitih tkiva. Voltaža zrake povećava njenu energiju te generalno smanjuje kontrast. Korištenje kontrastnog sredstva povećava razliku i čini neka tkiva jasnije vidljivima.

3.1.4. Dijelovi srca

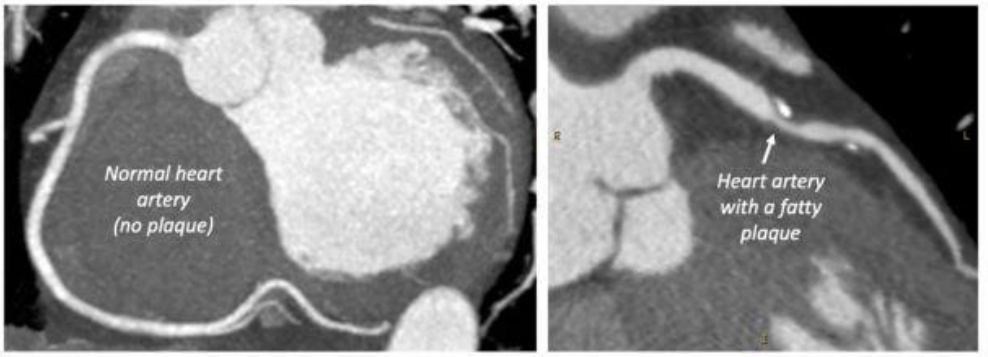
Srce je organ kardiovaskularnog sustava, u prosjeku veličine šake, čija je primarna svrha pumpanje krvi kroz tijelo (Sl. 3.4.). Sastoji se od 4 glavna dijela sastavljenih od mišića kontroliranih električnim impulsima. Mozak i živčani sustav upravljaju funkcijom srca. Osim pumpanja krvi, srce kontrolira i brzinu te ritam kucanja te održava konstantan krvni tlak u žilama. Locirano je u prednjem dijelu prsa s laganim nagibom na lijevu stranu prsnog koša te ga

štite rebra. Glavni dijelovi su lijeva klijetka i pretklijetka i desna klijetka i pretklijetka. Pretklijetke su postavljene iznad klijetki. Krv u srce stiže iz pluća prema desnoj pretklijetki nakon čega ide u desnu klijetku koja pumpa krv prema plućima kroz plućnu arteriju gdje se krv obogaćuje kisikom. Obogaćena krv iz pluća ide plućnim venama prema lijevoj pretklijetki nakon čega ide u lijevu klijetku te ona pumpa krv prema tijelu kroz veliku aortu i grana se na ostale arterije prema svim drugim organima. Opisana kruženja predstavljaju dva krvotoka: veliki – tjelesna cirkulacija i mali – pulmonalna ili plućna cirkulacija. Desna klijetka i pretklijetka zadužene su za primanje venske krvi i ponovo obogaćivanje kisikom dok su lijeva klijetka i pretklijetka zadužene za primanje obogaćene krvi i njeno slanje u ostatak dijela. Srčana pregrada između klijetki onemogućuje miješanje krvi desne strane ili venske krvi i krvi lijeve strane ili arterijske krvi. Između svake pretklijetke i klijetke nalaze se srčani zalisci koji se ponašaju poput ventila kako bi spriječili povrat krvi u srčane šupljine nakon prolaska. Oni se nalaze i na početku velikih krvnih žila poput plućne arterije i aorte.

Na CT slikama srca mogu se vidjeti koronarne arterije koje dovode krv srcu, srčane komore, mišići i zalisci, plućne vene, torakalna aorta, ponekad i abdominalna aorta te perikardium (Sl. 3.5.).



Sl. 3.5. Prikaz srčanih komora [12]



Sl. 3.4. CT slika normalnog srca (lijevo) i srčane arterije sa masnim naslagama (desno) [13]

3.2. U-net konvolucijska neuronska mreža

U-net je arhitektura konvolucijske neuronske mreže za brzo i precizno semantičko segmentiranje slika koje se izvodi u području računalnog vida. Evoluirala je iz tradicionalne konvolucijske neuronske mreže te je dizajnirana u 2015. godini. Počela je biti šire korištena nakon što se njena performansa pokazala boljom od dotad korištenih rješenja (npr. sliding-window CNN).

3.2.1. Vrste segmentacije

U-net je razvijen kako bi se ostvarila semantička segmentacija slike visoke preciznosti. Osim semantičke segmentacije postoje i drugi tipovi segmentacija. U nastavku su ukratko objašnjena semantička segmentacija i drugi najvažniji tipovi segmentacija.

Semantička segmentacija uključuje dodjeljivanje klasa svakom pikselu slike. Modeli su trenirani koristeći segmentacijske mape koje predstavljaju ciljane varijable (*engl. target variables*). Cilj semantičke segmentacije je pružiti klasu koja na slici klasificira različite objekte koji su nam od interesa. U području obrade medicinskih slika to su područja različitih dijelova tijela, poput srčanih klijetki i slično. Rezultati segmentacije znaju biti teško definirani zbog čestih slučajeva preklapanja, dodirivanja i bliskog odnosa objekata koje treba različito kategorizirati. Zbog navedenog, često se koristi pred procesiranje ulaznih podataka kako bi se uklonili nedostatci s ulaznih slika.

Nadalje, osim semantičke segmentacije značajna je i segmentacija instance te panoptička segmentacija. Segmentiranje instance je segmentiranje u kojemu se pikseli klasificiraju bazirano na instancama objekta (za razliku od klasa). U ovom segmentiranju ne zna se kojoj klasi koje područje pripada nego se razdvajaju slična ili područja koja se preklapaju, a bazirana su na granicama objekata tako da bi se različiti objekti razdvojili i naglasila razlika između njih.

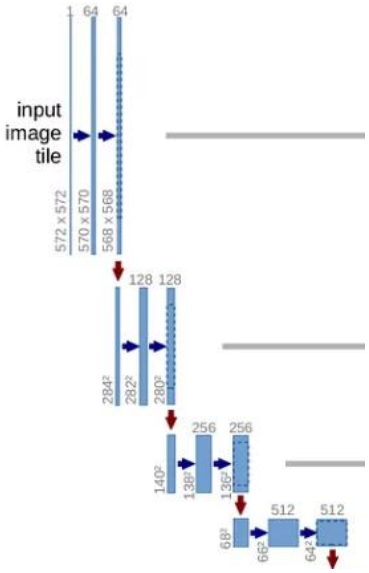
Panoptička segmentacija je segmentacija koja se može shvatiti kao kombinacija prethodno navedena dva tipa segmentacije. Panoptička segmentacija predviđa identitet svakog objekta odvajajući svaku instancu svakog objekta slike. Korisna je za mnoge slučajeve gdje se zahtjeva da se istovremeno obavlja obrada velike količine podataka. Praktičan primjer bi bio autonomno vozilo gdje bi auto trebalo istovremeno analizirati i razumjeti svoju okolinu brzo i precizno. To se može postići dovođenjem slika algoritmu panoptičke segmentacije.

3.2.2. Arhitektura

U-net arhitektura neuronske mreže sastoji se od nekoliko dijelova:

1. Enkoder – lijeva strana
2. Bottleneck – most koji spaja enkoder i dekoder
3. Dekoder – desna strana
4. Preskočne veze – poveznice između enkodera i dekodera

Arhitektura U-net mreže počinje enkoderom (Sl. 3.6.). Zadatak enkodera je kreirati kompaktne reprezentacije početne slike. Takve reprezentacije nižih dimenzija sadrže samo najvažnije informacije te slike. Drugim riječima osnovni zadatak enkodera je izdvojiti značajna svojstva sa



Sl. 3.6. Enkoder U-net neuronske mreže [1]

slike. Tim pristupom slika se sažima i svodi na najosnovnije i najnužnije podatke. Zato se ova grana U-net mreže zove grana sažimanja. Navedeno se postiže korištenjem konvolucijskih i pooling slojeva. Konvolucijski sloj je mapiranje ili kernel koji prolazi kroz svaki piksel slike. Ovo mapiranje je naučeno kroz proces treniranja modela. Konvolucijski sloj je osnovna komponenta konvolucijske neuronske mreže i on provodi većinu računanja. Konvolucijski sloj zahtjeva nekoliko komponenti, a to su: ulazni podatak, filter i mapa značajki (*feature map*). Filter ili kernel predstavlja detektor značajki koji se kreće po poljima ulaznog podatka, grayscale slike u ovom slučaju te provjerava je li prisutna značajka. Filter je dvodimenzionalna matrica težina koja predstavlja dijelove slike. Veličina filtera definira se visinom i širinom matrice filtera. U većini slučajeva koriste se manje kvadratne dimenzije poput 3x3 koje imaju težinu (*w*)

9. Nakon toga se filter primjenjuje na dio slike računanjem produkta između inputa piksela i filtera. Taj produkt se zatim sprema u izlaznu matricu nakon čega se filter pomakne ponavljajući proces dok ne prođe kroz cijelu matricu slike. Krajnji rezultat iz ulaza i filtera je matrica težina koja se naziva aktivacijskom matricom. Konvolucijski sloj na početku mreže uči prepoznati detalje i strukture poput rubova objekata na slikama. Što dulje u mrežu odlazi, on uči prepoznavati kompleksnije značajke poput geometrijskih oblika i lica pred kraj mreže.

Nakon toga se korištenjem predefinirane funkcije sažimanja tj *max pooling-a* smanjuju dimenzije ulazne slike. Slaganjem više slojeva konvolucije nakon koje slijedi *pooling* sloj, ekstraktira se više detalja informacije. Sloj sažimanja provodi dimenzionalnu redukciju smanjujući broj parametara u inputu. Slično kao konvolucijski sloj prelazi preko matrice cijelog inputa samo što nema nikakvih težina. Umjesto toga kernel primjenjuje agregacijsku funkciju veličinama unutar promatranog polja. Postoje 2 osnovna tipa sažimanja:

- Maksimalno – kako se filter kreće preko matrice inputa bira piksel s najvećom vrijednosti koju šalje izlaznom polju.
- Prosječno – kako se filter kreće preko matrice inputa računa se prosječna vrijednost u promatranom području kojeg šalje izlaznom polju.

Korištenjem sloja sažimanja smanjuje se kompleksnost, poboljšava efikasnost i limitira *overfitting*.

Preciznije gledano, enkoder se sastoji od dvije 3x3 konvolucije svaka praćena sa ReLU aktivacijskom funkcijom. ReLU unosi ne linearnost u mrežu koja pomaže u boljoj generalizaciji podataka za treniranje. Aktivacijska funkcija poznata kao ReLU omogućava brže i efikasnije treniranje mapiranjem negativnih vrijednosti i održavajući pozitivne. Zove se aktivacijska pošto samo aktivirane vrijednosti mogu nastaviti u sljedeći sloj. Vrlo je slična linearnoj aktivaciji pa ju je vrlo lako optimizirati. Za razliku od drugih nelinearnih aktivacijskih funkcija, nije zahtjevna za računanje, zbog čega je algoritam učenja brži. Važno je izabrati dobru stopu učenja kako bi se izbjeglo odumiranje tijekom algoritma učenja pri kojemu veliki gradijenti pri prolasku kroz ReLU mogu odumrijeti te se nikad više aktivirati.

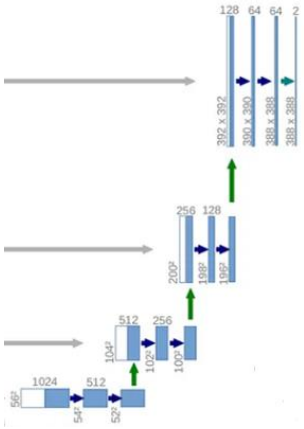
Izlazna vrijednost ReLU funkcije kasnije se koristi za preskočne veze (engl. *skip connections*) uz izlaz pripadajućeg bloka dekodera. Nakon ReLU aktivacijske funkcije slijedi 2x2 *max pooling* prethodno spomenut gdje se prostorna dimenzija slike upola smanjuje. Ovime se također

smanjuje trošak izvođenja smanjenjem broja parametara za treniranje. Svi ti dijelovi će biti prikazani u kodu u praktičnom dijelu.

Bottleneck označava dio U-net mreže koji se često svrstava polovično u enkoder i polovično u dekoder (Sl. 3.7). *Bottleneck* predstavlja most koji povezuje enkoder i dekoder i upotpunjuje tok informacije u mreži. Sastoji se od dvije 3x3 konvolucije gdje je svaka popraćena ReLU aktivacijskom funkcijom. Za razliku od dva susjedna dijela, nema nikakve kontrakcije ili ekspanzije dimenzije slike. Svrha ovog dijela je izdvojiti najbitnije detalje slike kako bi se mreža mogla fokusirati na bitne dijelove dok u isto vrijeme zaobilazi redundantne informacije. Prvi konvolucijski sloj ekstraktira lokalne značajke dok drugi pokušava pomoću temeljeno na prethodnom ekstraktirati kompleksnije i apstraktnije uzorke. Zbog ReLU funkcije ovdje se još dodatno utječe na ne linearnost mreže.

Sl. 3.7. Most/bottleneck U-net neuronske mreže [1]

Osim spomenutog, još jedan problem enkodera je u tome što je izlazna slika male dimenzije, puno manje u usporedbi s ulaznom. Ukoliko bi bilo korišteno za klasifikaciju, konačni sloj bi imao nekoliko čvorova, jedan za svaku klasu. Za segmentaciju, izlazna slika mora biti slika s istom visinom i širinom kao ulazna slika. U tu svrhu se uvodi dekoder (Sl. 3.8.). Svrha dekodera



Sl. 3.8. Dekoder U-net neuronske mreže [1]

je rekonstruirati sliku iz njene kompaktne reprezentacije. Isto kao kod enkodera, dekoder također ima konvolucijske slojeve koji su u ovom slučaju slojevi 2x2 dekonvolucije tj transponirane konvolucije koji povećavaju dimenziju slike. Kao što se *max pooling* koristi kako bi se slika sažela, tu istu svrhu samo suprotnu ima dekonvolucija. Dekonvolucijski sloj povećava dimenzionalnost koristeći funkciju koja je naučena, ne predefinirana. Tako da se funkcija za povećanje dimenzije update-a paralelno kako se model trenira. Enkoder je u mogućnosti predati važne informacije dekoderu, problem je što su lokacije značajki slike izgubljene. Način da se ovo riješi, odnosno da dekoder može naučiti točno rekonstruirati slike iz komprimiranih podataka je uvođenjem preskočnih veza koje se nakon transponiranja izvode *concatenate* funkcijom. Ta funkcija jedno je od najbitnijih značajki U-net arhitekture jer poboljšava performanse semantičke segmentacije i smanjuje količinu podataka koji je potreban za treniranje. Za autoenkodere, enkoder i dekoder moraju biti odvojeni kako bi se zadržala svrha kompresije slike, što nije slučaj u segmentaciji slike. U U-net mreži poveznice se koriste kako bi se predale informacije od prethodnog konvolucijskog sloja. Te informacije su lokacije značajki slike.

S obzirom na simetričnost mreže, dimenzije suprotnih slojeva će biti iste. Nakon ove funkcije izvode se dvije 3x3 konvolucija nakon koje slijedi ReLU funkcija, kao i u konvolucijskom sloju enkodera. Izlaz zadnjeg sloja dekodera dodatno prolazi kroz 1x1 konvoluciju najčešće sa sigmoidnom ili softmax aktivacijom u slučaju semantičke segmentacije. Sigmoidna aktivacija daje segmentacijsku masku koja predstavlja klasifikaciju na razini piksela te je korištena u ovom radu.

3.3. Metode predobrade

Predobrada igra bitnu ulogu u segmentaciju medicinskih slika jer povećava točnost i pouzdanost segmentiranih rezultata. S obzirom na to da se medicinske slike koriste u svrhe predviđanja i dijagnosticiranja bolesti, što veća točnost je ne samo poželjna nego ostavlja velik utjecaj na točnu dijagnozu. Segmentacija medicinskih slika uključuje identificiranje i izražavanje dijelova koji su od interesa unutar medicinske slike poput prepoznavanja tumora na MRI slici ili krvnih žila na CT angiogramu. Metoda predobrade ovisi o tome koje smetnje se najviše javljaju na slici te što je najbitnije za postići ovisno o dijagnostičkoj metodi (koja područja naglasiti ukoliko su samo određena područja promatrana ili poboljšati rezoluciju ukoliko su detalji jako bitni). Neki od češćih pristupa tome su smanjenje šuma, poboljšanje kontrasta, normalizacija i uklanjanje objekata.

Uklanjanje šuma je bitno jer se često pojavljuju razne smetnje na medicinskim slikama poput elektronskog šuma. Ovaj problem se najčešće rješava Gaussian ili Median filterima koji pomažu reducirati šum.

Poboljšanje kontrasta može pomoći naglasiti inače suptilne razlike u karakteristikama tkiva. U ove svrhe koriste se najčešće izjednačavanje histograma, CLAHE i *contrast stretching*.

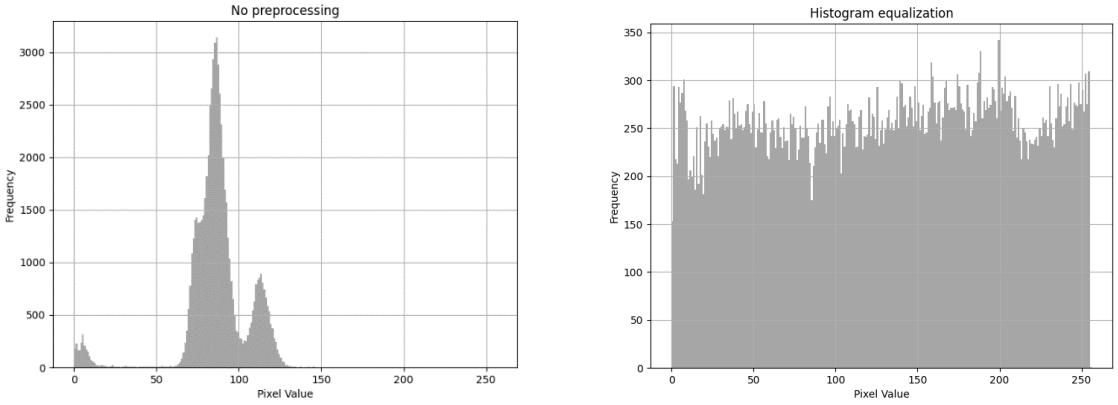
Normalizacija se koristi ukoliko se pojavljuju pikseli različitih intenziteta te se time osigurava konzistentnost vrijednosti kroz cijelu sliku.

Uklanjanje objekata sa slike koristi se u situacijama u kojim pacijenti imaju implantate ili kirurške instrumente koji ometaju rezultat segmentacije. Metode predobrade mogu detektirati i ukloniti te objekte.

S obzirom na korištene podatke za obradu i rezultat koji se želi postići u ovom slučaju se testira predobrada korištenjem metoda za poboljšanje kontrasta te uklanjanje šuma po potrebi.

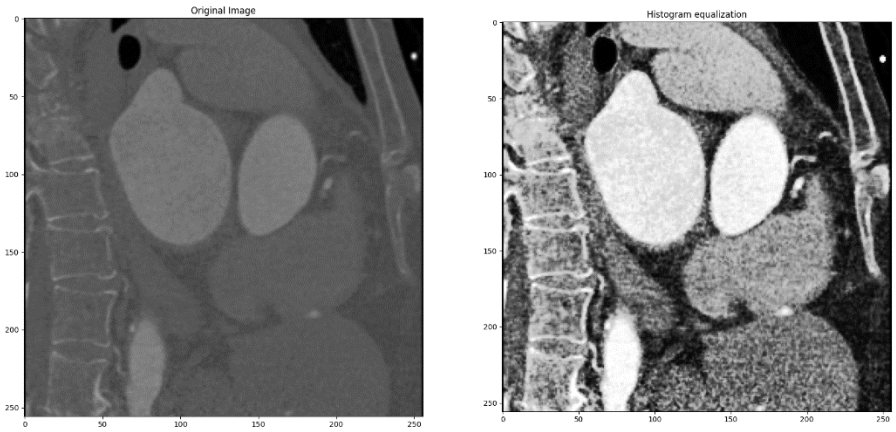
3.3.1. Izjednačavanje histograma

Histogram je grafička reprezentacija intenziteta distribucije slike. Prezentira broj piksela za svaki intenzitet slike. Izjednačavanje histograma je metoda procesiranja slika korištena kako bi se poboljšao kontrast. To postiže efektivno raspoređujući najčešće vrijednosti intenziteta tj. rastežući raspon intenziteta koji se pojavljuju na slici. Ovu metodu moguće je primijeniti i za slike u boji, ali ne radi se za svaki kanal zasebno (RGB format) jer bi to drastično promijenilo



Sl. 3.9. i 3.10. Histogram slike bez (lijevo) i slika sa izjednačavanjem histograma (desno)

sliku. Umjesto toga slika se može konvertirati u drukčiji format poput HSV te se tad ova metoda može primijeniti na prikladan kanal bez upotrebe drugih. Na slikama 3.9 – 3.12. može se vidjeti



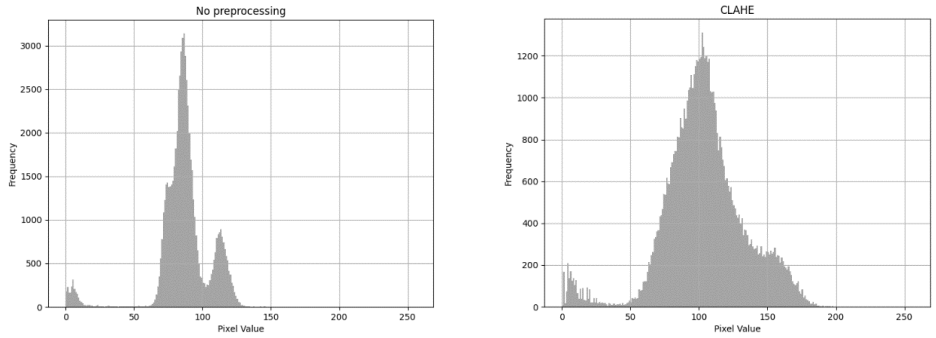
Sl. 3.11. i 3.12. Prikaz slike bez (lijevo) i slike sa izjednačavanjem histograma (desno) x presjeka

utjecaj ovog algoritma na izgled histograma te na samu sliku.

3.3.2. CLAHE

CLAHE je kratica za engleski naziv *Contrast Limited Adaptive Histogram Equalization* što bi otprilike značilo Adaptivno izjednačavanje histograma limitiranog kontrasta. Ono je verzija AHE tj. Adaptivnog izjednačavanja histograma. CLAHE je originalno kreiran algoritam kako bi se poboljšale medicinske slike slabog kontrasta. Sve dosad spomenute metode imaju svrhu mijenjanja kontrasta slike manipulacijom histograma slike. Običan AHE zna previše naglasiti područja s gotovo konstantnim kontrastom što rezultira u stvaranju šuma i stoga je napravljen CLAHE gdje se limitira pojačavanje kontrasta i to im je jedina razlika. Razlika između običnog izjednačavanja histograma i CLAHE-a je u tome što se u ovom slučaju računa nekoliko histograma, svaki za određeni dio slike koji se ne preklapa te se oni koriste kako bi se redistribuirala svjetlina slike zbog čega je ova metoda dobra za poboljšanje lokalnog kontrasta i naglašavanje rubova svake regije slike. Način na koji CLAHE rješava spomenut problem sa AHE je uvođenjem parametra *clipping limit* čime se prije računanja kumulativne funkcije distribucije limitira pojačanje tako da se histogram sreže na preferiranu vrijednost.

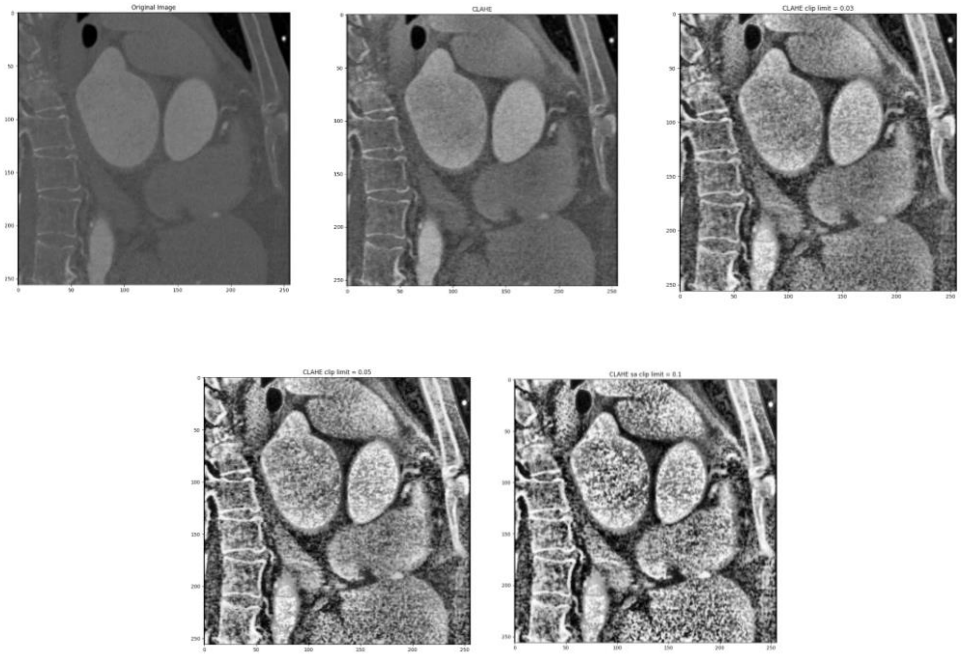
Na slikama 3.13. i 3.14..može se vidjeti kako CLAHE djeluje na histogram slike te sa različitim



Sl. 3.13. i 3.14. Histogram slike bez (lijevo) i slike sa CLAHE (desno)

clip limit-om.

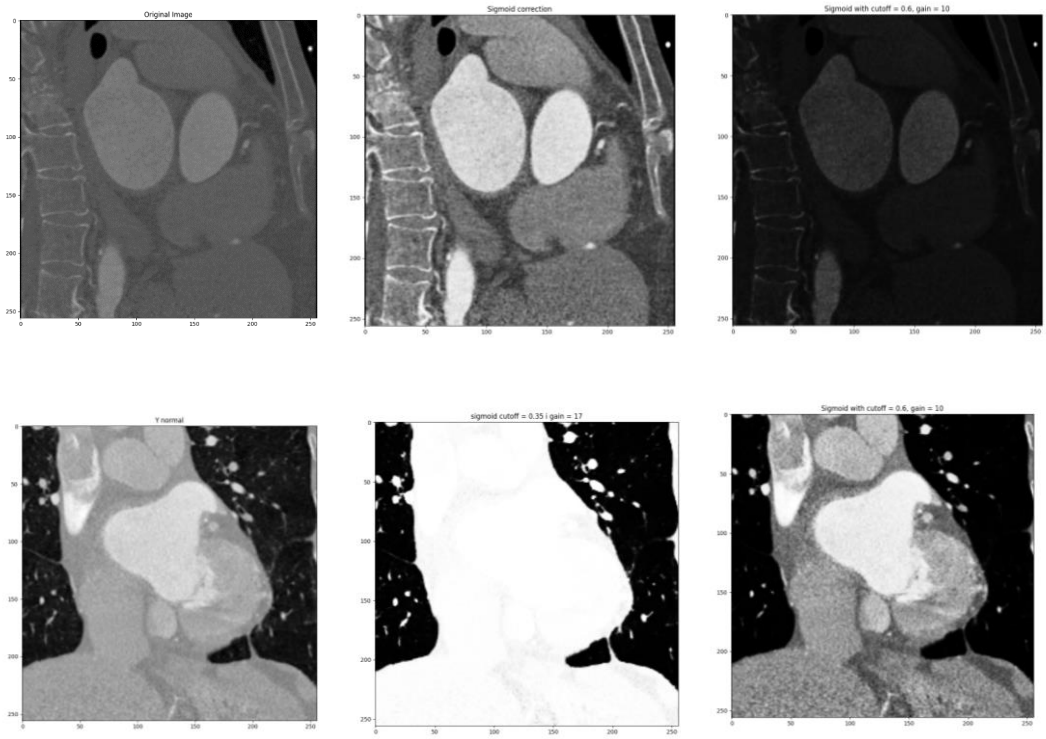
Na slikama 3.15.-3-19. vidi se kako primjena CLAHE utječe na sliku iz seta korištenog za treniranje. Također može se vidjeti kako utječe promjena parametra clip limit-a na izgled slike te zašto nije uzet veći broj. Za predobradu korištenu pri treniranju koristio se clip limit od 0.01 i 0.03, a na slikama je dodatno prikazana i slika sa 0.05 i 0.1. Vidi se da se povećanjem parametra povećava šum te da slika postaje sve manje prepoznatljiva.



Sl. 3.15. , 3.16. , 3.17. , 3.18. i , 3.19. Prvi red: prikaz slike bez predobrade (lijevo), sa CLAHE clip limit = 0.01 (sredina) i CLAHE clip limit = 0.03 (desno), drugi red: slika sa CLAHE clip limit = 0.05 (dolje desno) i CLAHE clip limit = 0.1 x presjeka

3.3.3. Sigmoid correction

Sigmoidna korekcija je noviji pristup koji koristi sigmoidnu funkciju. Ona je konstantna nelinearna funkcija „S“ oblika. Funkcija se primjenjuje direktno na svakom pikselu. U ovoj metodi maska koja se aplicira na sliku je ne linearna aktivacija koja predstavlja sigmoidnu funkciju pomnoženu s ulaznom slikom i zadanim faktorom. Faktor se koristi kako bi se odredio željeni stupanj kontrasta koji ovisi o svjetlini slike. Koristi se funkcija $\frac{1}{(1+e^{(gain(cutoff-I))})} = 0$ koja se primjenjuje na svaki piksel. *Gain* i *cutoff* se zadaju kao parametri funkcije dok *I*



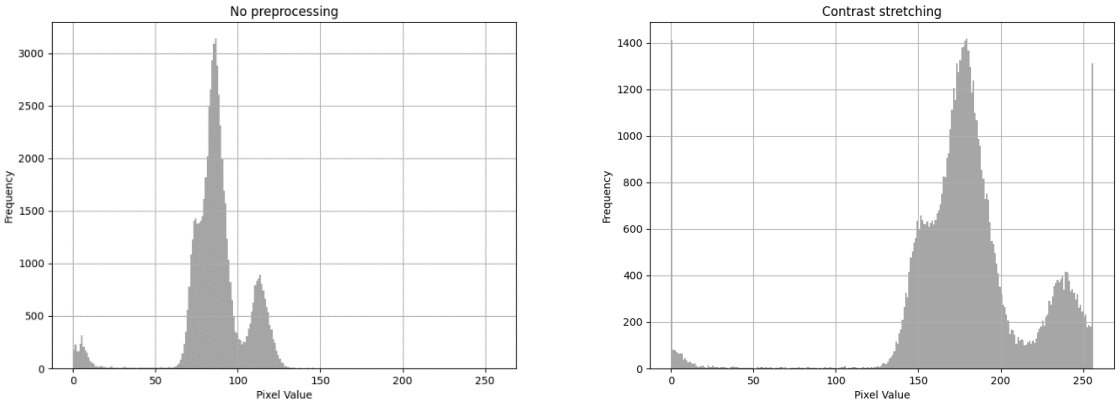
Sl. 3.20., 3.21., 3.22., 3.23., 3.24. i 3.25. Prvi red: slika bez predobrade (lijevo), sa sigmoid correction cutoff = 0.35 i gain = 17 (sredina), sa sigmoid correction cutoff = 0.6 i gain = 10 (desno) x presjeka, drugi red: slika bez predobrade (lijevo), sa sigmoid correction cutoff = 0.35 i gain = 17 (sredina), sa sigmoid correction cutoff = 0.6 i gain = 10 (desno) y presjeka

predstavlja vrijednost piksela. *Gain* kontrolira središte krivulje tj. infleksiju dok *cutoff* kontrolira nagnutost. U korištenim primjerima potrebno je namjestiti oba parametra prije treniranja kako slika ne bi bila pre tamna ili presvijetla. Dobra osobina ove metode je što ne uzrokuje šum na slici kao prethodne dvije metode, ali je zahtjevnija po tome što je potrebno dobro prilagoditi parametre kako bi se dobio dobar rezultat te se funkcija sama po sebi ne prilagođava dobro različitim slikama različitog intenziteta svjetlosti.

Na slikama 3.20 - 3.25.vidi se kako isti parametri ne odgovaraju slikama različitog intenziteta svjetline, ali ukoliko se parametri dobro namjeste dobije se dobra promjena kontrasta uz zadržavanje rubova i bez značajnog šuma na slici.

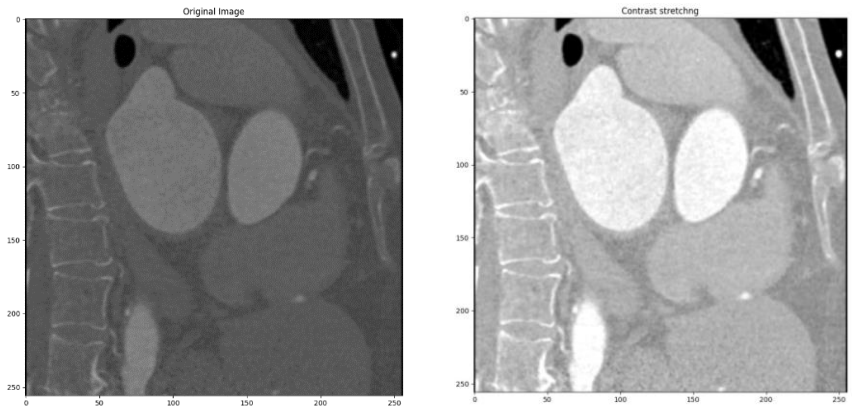
3.3.4. Contrast stretching

Contrast stretching je linearna metoda promjene kontrasta, često korištena u slikama kako bi se poboljšala vidljivost elemenata. Poboljšava kontrast tako što rasteže vrijednosti intenziteta slike da bi se popunio cijeli dinamički raspon koji je zadan. Transformacijska funkcija koja se koristi uvijek je linearna i raste. Formula za računanje je sljedeća: $Xnew = \frac{Xinput - Xmin}{Xmax - Xmin} x255$ gdje Xnew predstavlja novu vrijednost piksela, Xinput staru vrijednost, Xmin i Xmax zadane vrijednosti raspona unutar kojeg se želi dobiti vrijednost piksela. Mogu se zadati vrijednosti raspona kao konkretne vrijednosti piksela koje se očekuju, a može se zadati kao postotak koji



Sl. 3.26. i 3.27. Histogram slike bez (lijevo) i slike sa *contrast stretching* (desno)

označava postotak piksela trenutne slike. U tom slučaju se radi o *percentile stretching* dok se u prethodnom slučaju govorilo o *min-max stretching*. U *percentile stretching* za minimalnu i maksimalnu vrijednost koriste se pikseli unutar zadanog postotka tako da se maknu ekstremni pikseli iz jednadžbe. Ova metoda je superiornija i daje bolje rezultate te je korištena tijekom predobrade. U tom slučaju koristilo se *min* 2% i *max* 98% piksela. To se koristilo za sve primjere te se točna implementacija može naći u kodu u sljedećem poglavlju. Na slikama 3.26 -3.29. može se vidjeti utjecaj ove metode na medicinsku sliku korištenu za predobradu te kako je ova metoda utjecala na histogram slike.



Sl. 3.28. i 3.29. Prikaz slike bez (lijevo) i slike sa *contrast stretching* (desno) x presjeka

3.3.5. Median filter

Median filter je ne linearan filter u kojom se svaka izlazna vrijednost računa kao medijan ulaznih

Sl. 3.30. i 3.31. Prikaz slike bez (lijevo) i slike sa Median filterom (desno) x presjeka sa primijenjenim izjednačavanja histograma

vrijednosti. Koristi se najčešće za uklanjanje šuma sa slike i usput čuva rubove. Kao i Mean filter, Median filter gleda svaki piksel zasebno te usput gleda njegove susjede da odredi predstavlja li taj piksel svoje okruženje ili ne. Umjesto da jednostavno zamjeni piksel sa srednjom vrijednosti okolnih, zamjeni ga medijanom tih vrijednosti. Medijan se računa tako da se prvo sortiraju sve vrijednosti piksela susjeda numeričkim redoslijedom te se nakon toga mijenja središnji piksel s vrijednosti piksela koji se smatra središnjim. Ako se gleda paran broj

susjeda uzima se srednja vrijednost dva u sredini. Razlog zašto je ovdje izabran je jer on za razliku od Gaussian i Mean filtera ostvaruje dobre rezultate pri očuvanju rubova na slici što je vrlo bitno za ovu segmentaciju. Na slici 3.30. i slici 3.31 vidi se kako Median filter utječe na izgled slike na koju je prethodno primijenjen izjednačavanje histograma. Filter uklanja dio šuma te boja slike postane blago ravnomjernija s rubovima još uvijek dobro definiranima.

4. IMPLEMENTACIJA U-NET NEURONSKE MREŽE I PREDOBRADE

U ovom dijelu dan je opis razvijenog sustava za izvedbu segmentacije i predobrade. Programski jezik korišten za implementaciju je *Python* zbog velikog boja lako dostupnih biblioteka koje nude mnogo gotovih funkcija u području obrade slike i strojnog učenja. Korištena je verzija *Pythona* 3.11. koja je trenutno najnovija verzija. Proces izvođenja segmentacije pomoću U-net mreže te predobrade podataka provodi se prvo učitavanjem željenih podataka što su kao što je već spomenuto u ovom slučaju .jpg slike presjeka srca. Oni se učitavaju te ukoliko je to zadano, primjenjuje se funkcija predobrade. Nakon tog koraka slijedi definiranja modela koji se koristi.

Neke od korištenih funkcija su gotove te ih nude biblioteke dostupne unutar Python-a, ali cijela U-net mreža je složena od tih dijelova i nije gotova funkcija koja postoji. Nakon definicije model

```
import numpy as np
from skimage.transform import resize
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.layers import Input, Conv2D, Conv2DTranspose, MaxPooling2D
from tensorflow.keras.layers import concatenate
from tensorflow.keras.utils import img_to_array, load_img
from skimage.exposure import equalize_adapthist, equalize_hist, adjust_sigmoid, rescale_int
from skimage.filters import gaussian, median
```

Sl. 4.1. Kôd korištenih biblioteka

se kompajlira i trenira na učitanim podatcima. Nakon čeka se podatci prikazuju i spremaju te po potrebi dodatno obrađuju. Na slici 4.1. prikazane su biblioteke korištene u kodu.

4.1. Učitavanje podataka

Dostupan broj slika unutar ulaznog skupa podataka je 20 .nii slika koje se pretvorbom u .jpg pretvore u 20 datoteka s u svakoj po još 3 datoteka u kojima je 512 (x os), 512 (y os) ili 363 (z os) slika. U ovom slučaju mreža se trenirala s manje zbog nemogućnosti komercijalnih računala da treniraju na tolikoj količini podataka te kako bi se bolje izrazila razlika u utjecaju metoda predobrade s obzirom na to da je mreža trenirana na puno podataka davala odlične rezultate same po sebi te je bilo zahtjevnije zamijeniti razliku sa predobradom. Za učitavanje slika koristile su se metode iz *Tensorflow*-a jer su se pokazale najboljima i davale najbolje rezultate. Iako druge biblioteke nude učitavanje podataka u istom formatu, postoji razlika u performansi i zbog toga je

izabrana ova biblioteka. Slike su skalirane na dimenziju 256x256 jer nisu kvadratne što stvara problem kasnije funkciji *concatenate* te je također ovo jedna od preporučenih veličina za treniranje zbog načina rada mreže. Nakon definiranja veličine slike, definira se količina slika koja ovisi o osi te se definira broj korištenih datoteka. Ti brojevi su bitni za funkciju učitavanja i za stvaranje polja koju učitava slike. Slike će se nalaziti u polju elemenata numpy funkcije te se polje u početku definira s nulama i prethodno zadanim veličinama.

```
#set img size (this is prefered)
im_width = 256
im_height = 256

num_img = 512 # or 363 for different axis (x - 512, y - 512, z - 363)
num_files = 1 # depending of number of different image files used - 20 is maximum

x = np.zeros((num_img*num_files, im_height, im_width, 1), dtype=np.float32)
y = np.zeros((num_img*num_files, im_height, im_width, 1), dtype=np.float32)
```

Sl. 4.2. Kôd za definiranje polja i veličine slike

Funkcija učitavanja slika u polje (engl. *array*) zove se *get_inputs* i ne prima parametre. U njoj se izvršavaju 2 petlje koje označavaju količinu slika u datoteci i količinu datoteka kojih se uči. Ovo se izvodi za svaki presjek zasebno radi jednostavnosti i brzine programa. Nakon učitavanja slike, slika se pretvara u polje elemenata pomoću funkcije *img_to_array* te se nakon toga promijeni veličina funkcijom resize s vrijednostima prethodno zadanima. Zadnji korak prije spremanja u polje je normaliziranje vrijednosti na između [0,1] dijeljenjem instance sa 255. Nakon toga slijede isti koraci, samo za set izlaznih slika. Ukoliko se dodaje predobrada, potrebno je još dodati željenu funkciju za predobradu prije spremanja u polje. Funkcije za predobradu koriste se iz *skimage.exposure* zbog najbolje kompatibilnosti s metodama za učitavanje podataka. Zbog sposobnosti Pythona da funkcija vraća više vrijednosti tako je ova funkcija u mogućnosti vratiti x i y što predstavljaju ulazne i izlazne slike.

```
# uncomment to apply desired processign method
def get_inputs():
    for i in range(1, (1 + num_files)):
        for j in range(0, num_img):
            img = load_img("med2image-master/outOr_"+ str(i) +"/x/output-slice"+ str(j)+ ".jpg", color_mode = "grayscale")
            x_img = img_to_array(img)
            x_img = resize(x_img, (256, 256, 1), mode = 'constant', preserve_range = True)
            mask = load_img("med2image-master/outTr_"+ str(i) +"/x/output-slice"+ str(j)+ ".jpg", color_mode = "grayscale")
            mask = img_to_array(mask)
            mask = resize(mask, (256, 256, 1), mode = 'constant', preserve_range = True)
            x[j + ((i-1)*num_img)] = x_img/255.0

            #x[j + ((i-1)*num_img)] = equalize_hist(x[j + ((i-1)*num_img)])
            #x[j + ((i-1)*num_img)] = equalize_adapthist(x[j + ((i-1)*num_img)], clip_limit = 0.03)
            #x[j + ((i-1)*num_img )] = adjust_sigmoid(x[j + ((i-1)*num_img)], cutoff= 0.35, gain = 17)
            ##0.35 i 17 za x, 0.6 i 10 za y 0.35 i 10 za z
            #p2, p98 = np.percentile(x[j + ((i-1)*num_img)], (2, 98))
            #x[j + ((i-1)*num_img)] = rescale_intensity(x[j + ((i-1)*num_img)], in_range=(p2, p98))
            #x[j + ((i-1)*num_img)] = median(x[j + ((i-1)*num_img)])

            y[j + ((i-1)*num_img)] = mask/255.0

    return x, y
```

Sl. 4.3. . Kôd za definiranje funkcije za učitavanje slika

```
x, y = get_inputs()

x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size = 0.1, random_state = 12)
```

Sl. 4.4. Kôd za pozivanje funkcije učitavanja i podjele skupa na skup za treniranje i skup za validaciju

Nakon poziva funkcije i spremanja u prethodno definirane strukture podataka, ti podatci se dijele na set za treniranje i set za validaciju pomoću funkcije *train_test_split* koja nasumično bira slike za oba seta te se trenira na 90% i testira na 10% slika. Parametar *random_state* je korišten kako bi se uvijek dobio isti set nasumičnih slika te kako bi rezultati bili konzistentniji. Taj broj je proizvoljan i može se uzeti bilo koji u određenom ponuđenom rasponu. Nakon ovih koraka prelazi se na definiranje i treniranje modela.

4.2. U-net

U-net se u ovoj izvedbi sastoji od dvije definirane funkcije. To su generalna i najbitnija funkcija *get_unet* koja vraća konačan model i *conv2d_block* koja predstavlja često ponavljajući blok konvolucija 3x3 svaka sa Relu aktivacijom koji se poziva često u *get_unet* funkciji. Funkciju *get_unet* nije potrebno pojednostavniti više od ovog s obzirom na to da je potrebno imati sve vrijednosti koje se vraćaju (jer se koriste u kasnijim dijelovima mreže, u već spomenutim preskočnim vezama). Na slici ispod se nalazi kod *conv2d_block* funkcije.

```
# 3x3 conv block
def conv2d_block(input_tensor, n_filters):
    # first layer
    conv = Conv2D(filters = n_filters, kernel_size = (3, 3), activation = 'relu',
                  kernel_initializer = 'he_normal', padding = 'same')(input_tensor)

    # second layer
    conv = Conv2D(filters = n_filters, kernel_size = (3, 3), activation = 'relu',
                  kernel_initializer = 'he_normal', padding = 'same')(input_tensor)

    return conv
```

Sl. 4.5. Kôd za definiranje funkcije za konvolucijski blok

Funkcija *conv2d_block* vraća konvolucijski sloj te se koristi u glavnoj funkciji. Prima samo parametre *input_tensor* i *n_filters* gdje *input_tensor* označava sliku koja prolazi kroz sve potrebne slojeve, a *n_filters* je parametar koji se u U-net mreži povećava kako se povećava dubina mreže, ali može se i treba namještati ovisno o potrebama. Bitno je da je taj parametar isti za ovaj konvolucijski blok kako bi mreža zadržala simetričnost. *N_filters* omogućava kontrolu kapaciteta i kompleksnosti modela. Veća vrijednost može zapaziti kompleksnije značajke, ali može povećati računalne potrebe i riskira overfitting. Česta je praksa, tako i ovdje što se vidi u kodu na slici 4.6. za u-net mrežu da se mijenja broj *n_filters*. Broj raste kako slika postaje manja kako bi bolje mogla zapaziti značajke. U početku se radi o detaljima niske razine poput rubova i teksture i s većom kompleksnosti dolazi se do detalja više razine poput prepoznavanja željenog objekta u odnosu na pozadinu. Isto tako vrijedi i za drugi dio U-net mreže, kako se povećava dimenzija tako se smanjuje broj filtera. To pomaže rasteretiti mrežu od nepotrebne kompleksnosti te također pridonosi simetriji mreže i pomaže održati istu dimenziju potrebnu za preskočne veze. Unutar funkcije *conv2d_block* nalaze se samo dva poziva funkcije *Conv2D* koja se poziva iz keras.layers. Zbog građe U-net mreže uvijek je kernel isti (3x3). Također jedan od parametara je i spomenuta ReLU aktivacijska funkcija. Parametar *kernel_initializers*, specifično korišten ovdje *he_normal* tipičan je parametar koji prati ReLu funkciju te specificira kako su inicijalne vrijednosti težina filtera postavljene. Parametar padding utječe na kako se konvolucijska mreža nosi s rubovima ulazne slike. Parametar „*same*“ označava da je na ulaznu sliku dodan rub vrijednosti 0 koji ima svrhu očuvanja prostorne dimenzije tijekom konvolucije.

Get_unet funkcija prikazana je na na slici 4.6.

```
# defined u-net model for training
def get_unet(input_img, n_filters = 32):

    # contracting path
    c1 = conv2d_block(input_img, n_filters * 1)
    p1 = MaxPooling2D((2, 2))(c1)

    c2 = conv2d_block(p1, n_filters * 2)
    p2 = MaxPooling2D((2, 2))(c2)

    c3 = conv2d_block(p2, n_filters * 4)
    p3 = MaxPooling2D((2, 2))(c3)

    c4 = conv2d_block(p3, n_filters * 8)
    p4 = MaxPooling2D((2, 2))(c4)

    # bottleneck
    c5 = conv2d_block(p4, n_filters = n_filters * 16)

    # expansive path
    u6 = Conv2DTranspose(n_filters * 8, (3, 3), strides = (2, 2), padding = 'same')(c5)
    u6 = concatenate([u6, c4])
    c6 = conv2d_block(u6, n_filters * 8)

    u7 = Conv2DTranspose(n_filters * 4, (3, 3), strides = (2, 2), padding = 'same')(c6)
    u7 = concatenate([u7, c3])
    c7 = conv2d_block(u7, n_filters * 4)

    u8 = Conv2DTranspose(n_filters * 2, (3, 3), strides = (2, 2), padding = 'same')(c7)
    u8 = concatenate([u8, c2])
    c8 = conv2d_block(u8, n_filters * 2)

    u9 = Conv2DTranspose(n_filters * 1, (3, 3), strides = (2, 2), padding = 'same')(c8)
    u9 = concatenate([u9, c1])
    c9 = conv2d_block(u9, n_filters * 1)

    # end of model
    outputs = Conv2D(1, (1, 1), activation = 'sigmoid')(c9)
    model = Model(inputs=[input_img], outputs=[outputs])

    return model
```

Sl. 4.6. Kôd za definiranje funkcije kreiranja U-net modela

Parametri koje navedena funkcija prima su ulazna slika i *n_filters*. Početni broj *n_filters* se može mijenjati ovisno o potrebama mreže, a u ovom slučaju se koristio početni broj 32 jer je pokazivao najbolje rezultate. Funkcija počinje s kontrakcijskim dijelom ponavljajući isti blok 4

puta te svaki poziv spremajući u zasebnu varijablu. Poziva se prethodno spomenuta definirana funkcija *conv2d_blok* nakon čega se poziva *MaxPooling2D* također iz *keras.layers* koji služi za smanjivanje dimenzije slike. *MaxPooling2D* ima samo parametar koji je tipičan za u-net arhitekturu. Broj filtera se povećava 2 puta u svakom bloku. To povećanje nije nužno, ali može pomoći mreži, ali je bitno imati iste brojeve na bloku ekspanzivnog dijela mreže zbog preskočnih veza. U najdonjem dijelu mreže nalazi se jedan poziv funkcije *conv2d_block* te se također mijenja veličina parametra *n_filters* pošto se radi o sloju s najmanjom dimenzijom slike gdje se promatraju najkompleksnije značajke. Nakon toga slijedi povećavanje slike gdje se radi o također 4 bloka. Poziva se *Conv2Dtranspose* što predstavlja dekonvoluciju tj povećavanje dimenzije slike. Parametar strides isti je kao u *MaxPooling2D* te je dobro da te vrijednosti budu jednake kako bi se mogle koristiti preskočne veze. Osim toga koristi se i parametar *n_filters* koji se postepeno smanjuje te padding koji ima istu ulogu i ponašanje kao prethodno spomenuto. Nakon toga slijedi concatenate funkcija koja izvršava ulogu preskočnih veza povezujući kontrakcijski put da ekspanzivnim. Kao parametar prima samo odgovarajuće varijable. Na kraju bloka dolazi konvolucijski blok s istim tipom parametara te istim brojem *n_filters* kao i *Conv2DTranspose*. Ti blokovi se ponavljaju sve do kraja kada se poziva još jednom funckija konvolucije te se predaje parametar za sigmoidnu aktivaciju. Konvolucija također više nije 3x3 nego 1x1 zbog arhitekture U-net mreže. Na samom kraju krajnja dobivena varijabla predaje se

```
# define inputs for model
input_img = Input((im_height, im_width, 1), name = 'img')

model = get_unet(input_img, n_filters = 32)

# compile model
model.compile(optimizer = Adam(), loss = "binary_crossentropy", metrics = ["accuracy"])
```

Sl. 4.7. . Kôd za stvaranje i kompajliranje modela

funkciji model iz keras.models koja stvara instancu tog tipa koji je korišten u daljnjim koracima.

Prije poziva funkcije *get_unet* instancira se keras tenzor pomoću funkcije *Input()* koja se poziva iz keras.layers. Keras tenzor je objekt kojemu se dodjeljuju parametri koje želimo da ulazni podatci za treniranje imaju te se koristi kako bi se izgradio Keras model kojemu će se u procesu treniranja dodijeliti pravi input. Nakon instance tenzora poziva se prethodno definirana funkcija i

predaje joj se zajedno s parametrom *n_filters*. Nakon definiranja instance modela model se kompajlira birajući optimizer, funkciju gubitka i metriku. Mogu se koristiti razni optimizatori, a Adam je ovdje izabran jer je općenito korišten za slične modele. Funkcija gubitka je metoda koja se koristi za mjerenje razlike između predviđene i stvarne vrijednosti. Ona ovisi o tipu segmentacije. Za binarnu segmentaciju često se koristi *'binary_crossentropy'*, a za višeklasnu segmentaciju koristi se *'categorical_crossentropy*'. U ovom radu korišteno je *'binary_crossentropy'* pošto nema dodjeljivanje klasa nego samo jedna očekivana izlazna vrijednost. Za specifične zadatke može se kreirati i vlastita funkcija gubitka. Za metriku je

```
model_name = 'model.h5' # define model name where to save
logs_name = 'logs.log' # set logs file name

# callbacks to save logs, define additional behaviour and set name
callbacks = [
    EarlyStopping(monitor='val_loss', patience = 8, restore_best_weights = True),
    ReduceLROnPlateau(factor = 0.1, patience = 5, min_lr = 0.00001, verbose = 1),
    ModelCheckpoint(model_name, verbose = 1, save_best_only = True),
    CSVLogger(logs_name)
]

# model training
results = model.fit(x_train, y_train, validation_data = (x_valid, y_valid),
                    batch_size = 32, epochs = num_epochs, callbacks = callbacks)
```

Sl. 4.8. . Kôd za definiranje callbacks i treniranje modela

izabran *'accuracy'* zbog široke uporabe i lakog razumijevanja iako će se rezultati najbolje procijeniti vizualno jer je teško procijeniti kvalitetu rezultata segmentacije metrički.

Sljedeća bitna stavka je definiranje *callbacks*. Navedeno nije bitno ukoliko se model ne želi trenirati s više epoha ili se ne želi pratiti tijek treniranja. Ovdje se definira ranije prekidanje treniranja pomoću *EarlyStopping()* gdje se određuju uvjeti i strpljenje. Prati se validacijski gubitak i ukoliko se on ne promijeni unutar 8 epoha prekida se treniranje. Funkcija *ReduceLROnPlateau* se koristi za poboljšanje stabilnosti neuronske mreže, koristi se tijekom treniranja kako bi dinamički promijenila (smanjila) stopu učenja optimizera pri određenim uvjetima. Također se može definirati i minimalna stopa učenja kako ne bi pala ispod određenog broja. Može se specificirati metrika koja se prati, ali ukoliko je prethodno definirana u funkciji prije nje, nije ju potrebno opet definirati, a isto vrijedi i za parametar patience. Funkcija *ModelCheckpoint* ima ulogu spremanja najboljeg modela u ovom slučaju. *CSVLogger* sprema log-ove treniranja kako bi se moglo pratiti treniranje. Radi nemogućnosti treniranja na svim

dostupnim podatcima uzet je veći broj epoha kako bi se dobili sličniji rezultati s manje podataka. Povećavanje epoha utječe na preciznost i na konačni rezultat. Ovdje je izabran broj epoha 25 kako bi se dobio željeni rezultat. Nedostatak korištenja većeg broja epoha je duže izvođenje programa. Taj broj se definira tijekom funkcije za treniranje modela model.fit. Toj funkciji se još predaju podatci za treniranje i podatci za validaciju tj podatci koji će se koristiti za predviđanje, ali u ovom slučaju koriste se za mjerenje metrike modela tijekom treniranja. Također se predaje parametar prethodno definiranih callbacks i parametar batch_size koji određuje veličinu podataka nakon kojih će se model ažurirati.

Nakon svih koraka jedino je preostalo testiranje modela pomoću funkcije predict gdje se predviđaju vrijednosti za zadan skup podataka.

```
preds_val = model.predict(x_valid, verbose = 1)
```

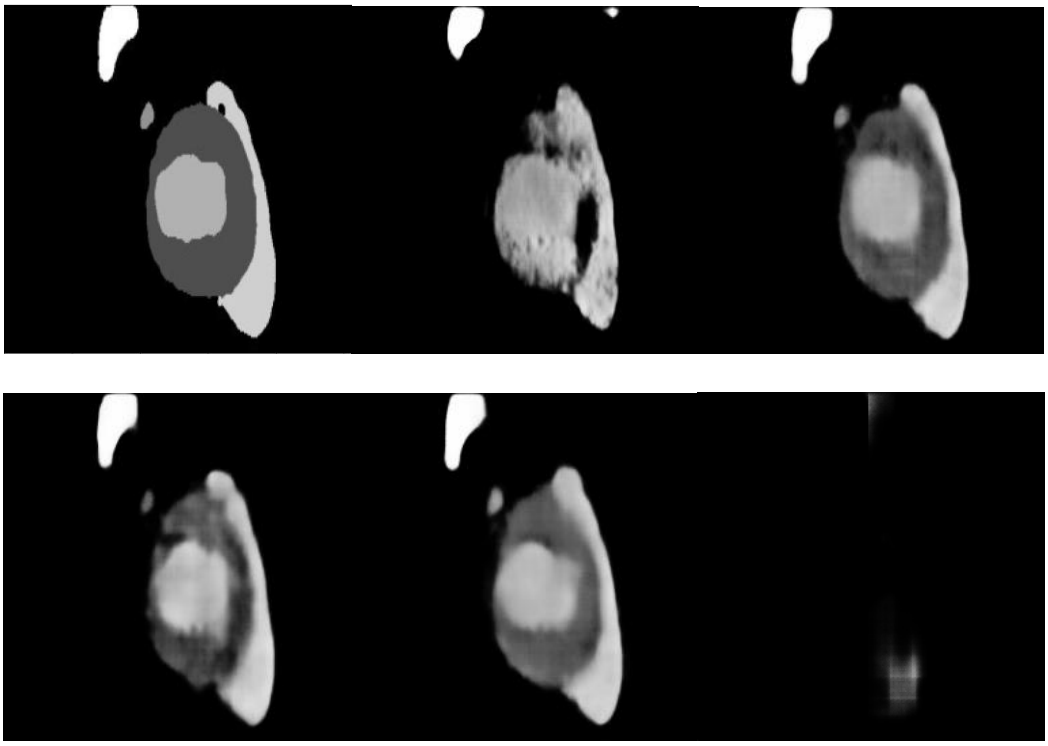
Sl. 4.9. . Kôd za predikciju korištenjem treniranog model

5. REZULTATI

U ovom dijelu rada predstavljeni su rezultati dobiveni generiranjem prethodno prikazanog modela korištenog za segmentiranje danog seta slika. Koristit će se nekoliko primjera, a glavni fokus je prikaz rezultata s metodama promjene kontrasta i rezultata uz dodatak korištenja dodatnog filtera za šum. Bit će prikazani modeli koji su trenirani na različitom broju slika te s različitim brojem epoha ovisno što se želi prikazati. Modeli koji su trenirani na više epoha bolje će prikazati promjene u detaljima, ali će biti trenirani na manjem broju slika zbog vremena koji je bio potreban za njihovo treniranje. Modeli koji su trenirani na manje epoha, ali s više slika bolje će pokazati utjecaj metoda za promjenu kontrasta te problematiku koja dolazi uz veći broj podataka za treniranje, dok će modeli trenirani s više epoha, ali manje slika bolje pokazati utjecaj metoda za promjenu kontrasta uz korištenje filtera za šum. Na taj način su podijeljena iduća poglavlja uz prikaz izgleda različitih presjeka. Metode koje će se koristiti su prethodno spomenute i opisane, ali će biti dodatno naglašeno o kojima se radi.

5.1. Usporedba metoda promjene kontrasta

U ovom dijelu komentirane su slike dobivene treniranjem modela s 10 različitih 3D slika što znači ukupno 5120 2D slika s početnom dimenzijom 512x200. Korišten broj epoha je 10 dok su svi ostali parametri isti kao u ostalim verzijama treniranja te se mogu pogledati u kodu na slici 4.8. priloženom u poglavlju 4. Ovdje su prikazani rezultati treniranja bez predobrade, s algoritmom izjednačavanja histograma, CLAHE s parametrom clip limit vrijednosti 0.01 i CLAHE s parametrom clip limit u vrijednosti od 0.03 i sigmoid correction sa parametrima cutoff = 0.35 i gain = 17. Iz rezultata prikazanim na slikama 5.1 - 5.5 vidi se da je model bez predobrade uspio odrediti pozicije nekih elemenata, ali nije ih potpuno detektirao te su boje većinski vrlo loše. Dok je s druge strane korištenjem izjednačavanja histograma poprilično povećana detekcija objekta te je čak uspješno detektirana i boja u područjima. Boja nije jednolična u tamno sivim područjima što je potencijalno rezultat količine šuma na slici. Rezultat korištenja CLAHE (clip limit = 0.01) je još uvijek dobar spram rezultata bez predobrade, ali lošiji spram prethodne metode predobrade. Povećanjem parametra clip limit poboljšan je rezultat te je ovaj put model bolje mogao detektirati rubove. U ovom slučaju povećanje kontrasta u CLAHE je pomoglo boljoj detekciji. Na primjeru sigmoid correction rezultat je neprepoznatljiv te nije bilo moguće odraditi detekciju s obzirom na to da se radi o većem broju različitih slika različitog intenziteta svjetlosti. U ovom slučaju izabrana je slika s velikim intenzitetom svjetline



Sl. 5.1. , 5.2. , 5.3. , 5.4. i , 5.5. Prvi red: očekivan rezultat (lijevo), rezultat bez predobrade (sredina), sa izjednačavanjem histograma (desno), drugi red: rezultat sa CLAHE clip limit = 0.01 (lijevo) i sa CLAHE clip limit = 0.03 (sredina) i sa sigmoid correction cutoff = 0.35 i gain = 17 (desno) x presjeka

što je rezultiralo u gotovo bijeloj slici i dovelo do loše predikcije. Ovaj rezultat je namjerno izabran kako bi se pokazao problem u korištenju ove metode predobrade. Utjecaj intenziteta svjetline prikazan je na sljedećim primjerima. Treniranje ovih modela obično traje nekoliko sati. Dužina treniranja se povećava najviše s povećanjem seta treniranja i povećanjem broja epoha.

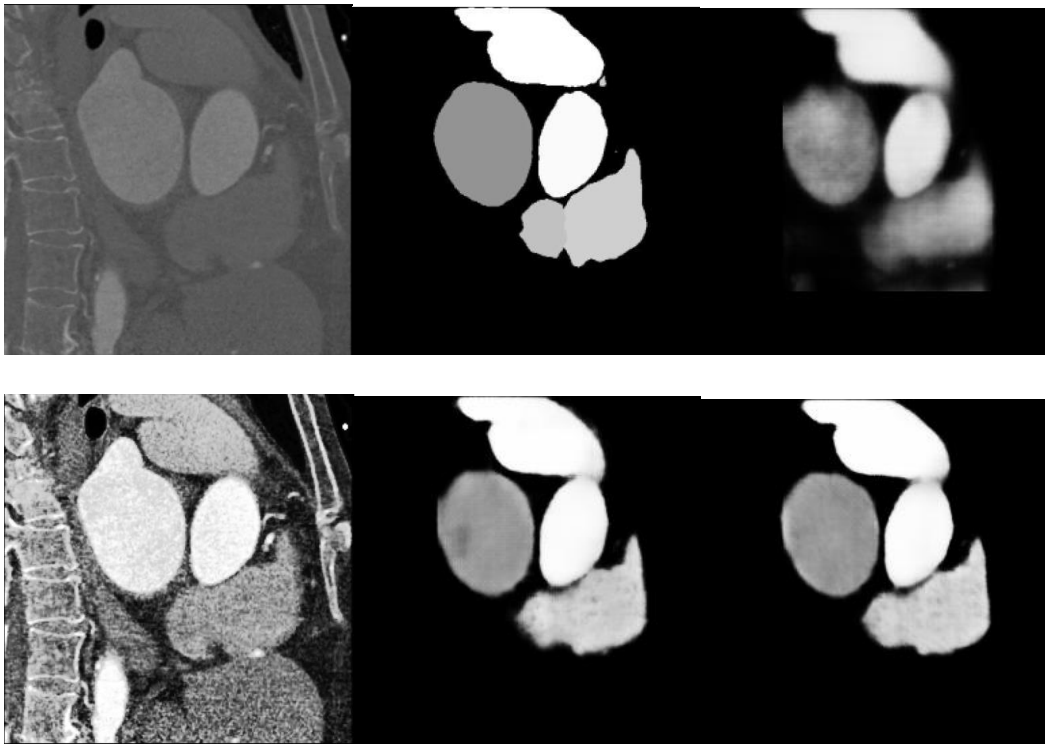
5.2. Usporedba metoda promjene kontrasta uz Median filter

Ovo potpoglavlje podijeljeno je na 3 dijela ovisno o presjeku. Komentirana je zasebno svaka metoda te na kraju svakog dijela komentiran je općeniti rezultat. U svakom skupu slika prikazan je dio povezan s modelom bez predobrade i s predobradom kako bi se mogao procijeniti utjecaj metode predobrade na konačan rezultat.

5.2.1. X presjek

Na slikama 5.6 - 5.11 vidi se da je u usporedbi s prethodnim slučajem broj epoha utjecao da rezultat bude puno jasniji i bliži željenom. U ovom slučaju korišten je algoritam izjednačavanja histograma uz dodatak korištenja median filtera. Slika bez predobrade daje naznake oblika i boja određenih područja, ali nedostaje dobro definiranih obruba dijelova srca te oblici nisu potpuni.

Na slici 5.10. s primjenom izjednačavanja histograma vidi se poboljšanje u definiranju oblika te su oblici puniji i jasniji. Nepravilnosti u oblicima su još uvijek prisutni i jasniji u nekim područjima, ali su bolji nego bez predobrade. U primjeru s dodatnom primjenom median filtera vidljivo je poboljšanje u jednoličnosti boje za zasebne dijelove te blagi utjecaj na rubove. U ovom slučaju na određenim mjestima slike je djelomično poboljšano definiranje rubova.

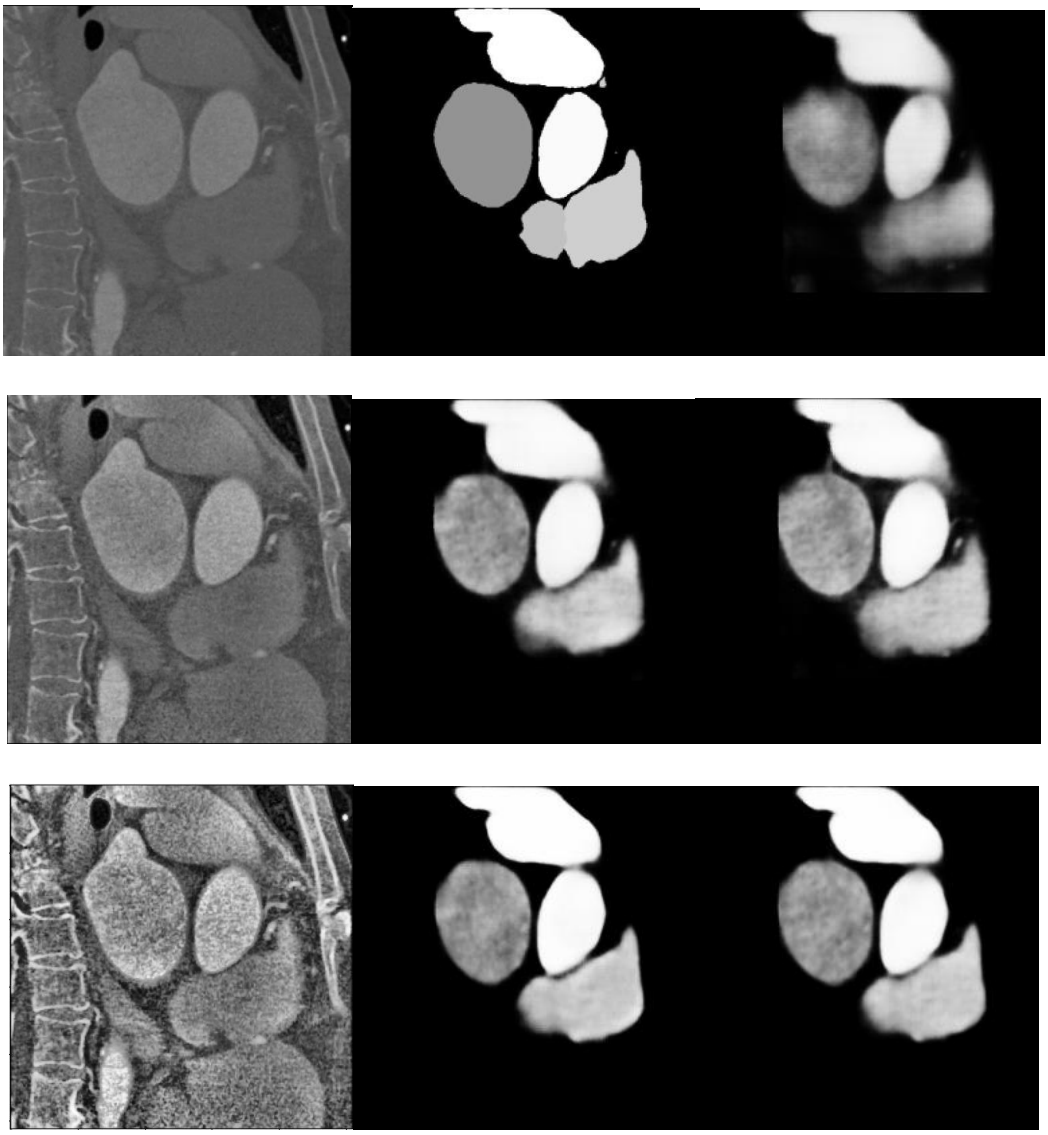


Sl. 5.6., 5.7., 5.8., 5.9., 5.10. i 5.11. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa izjednačavanjem histograma (lijevo), rezultat sa izjednačavanjem histograma (sredina), sa izjednačavanjem histograma i median filterom (desno) x presjeka

Na slikama 5.12. – 5.20. korišten je isti broj slika i epoha te algoritam predobrade CLAHE s dva različita parametra i uz korištenje median filtera. Na slici 5.16. gdje je korišten CLAHE (clip limit = 0.01) vidi se poboljšanje, ali rezultat još uvijek nema jasne obrube u nekim dijelovima slike. Na slici 5.19. gdje je korišten CLAHE (clip limit = 0.03) vidi se poboljšanje u odnosu na prethodnu sliku. Povećanje parametra pomoglo je u boljem definiranju obruba, ali boja je ostala jednake konzistentnosti. Korištenje median filtera na CLAHE (clip limit = 0.01) ostvareni su bolji rezultati prilikom definiranja rubova , ali nema značajnog poboljšanja u boji slike što može biti rezultat činjenice da slika s ovom vrijednosti clip limit-a nema značajan šum. S druge strane,

slika s većim clip limit-om dobiva vrlo mali utjecaj na određena područja (najtamnije područje slike dijela srca) gdje je blago zaglađeno područje dok je na drugim mjestima utjecaj

neprimjetan. Najbolji od priloženih rezultata je primjer gdje je korišten CLAHE (clip limit =

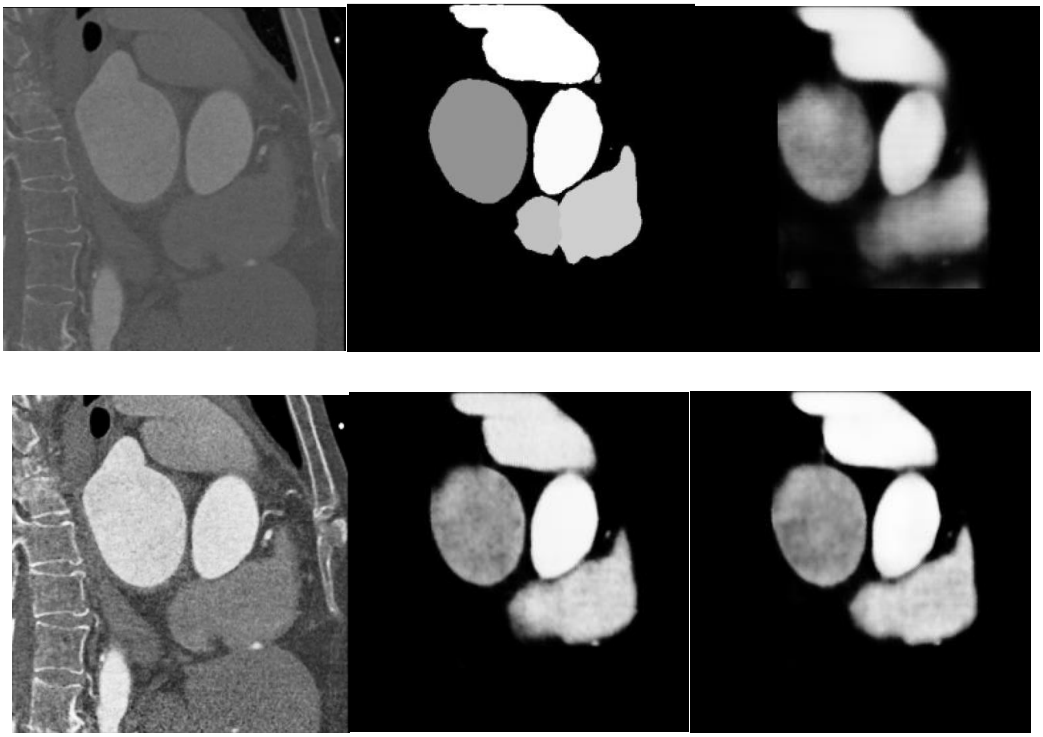


Sl. 5.12., 5.13., 5.14., 5.15., 5.16., 5.17., 5.18., 5.19. i 5.20. Prvi red: originalna slika (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa CLAHE clip limit = 0.01 (lijevo), rezultat sa CLAHE 0.01(sredina), sa CLAHE 0.01 i median (desno), treći red: original sa CLAHE clip limit = 0.03 (desno), rezultat sa CLAHE 0.03 (sredina), sa CLAHE 0.03 i median filterom (desno) z presjeka

0.03) te primjer sa CLAHE (clip limit = 0.03) popraćen median filterom. Za razliku od prethodnog primjera s izjednačavanjem histograma, ovdje median filter nema značajan utjecaj na rezultat jer se kontrast mijenja drukčije ovim algoritmom. Kontrast se više lokalizirano mijenja zbog čega se mijenja boja određenih područja. Dio koji je najsvjetliji je dobio više šuma u ovom slučaju te postao slabije prepoznatljiv. Vizualno procijenjeno čini se da je najbolji rezultat

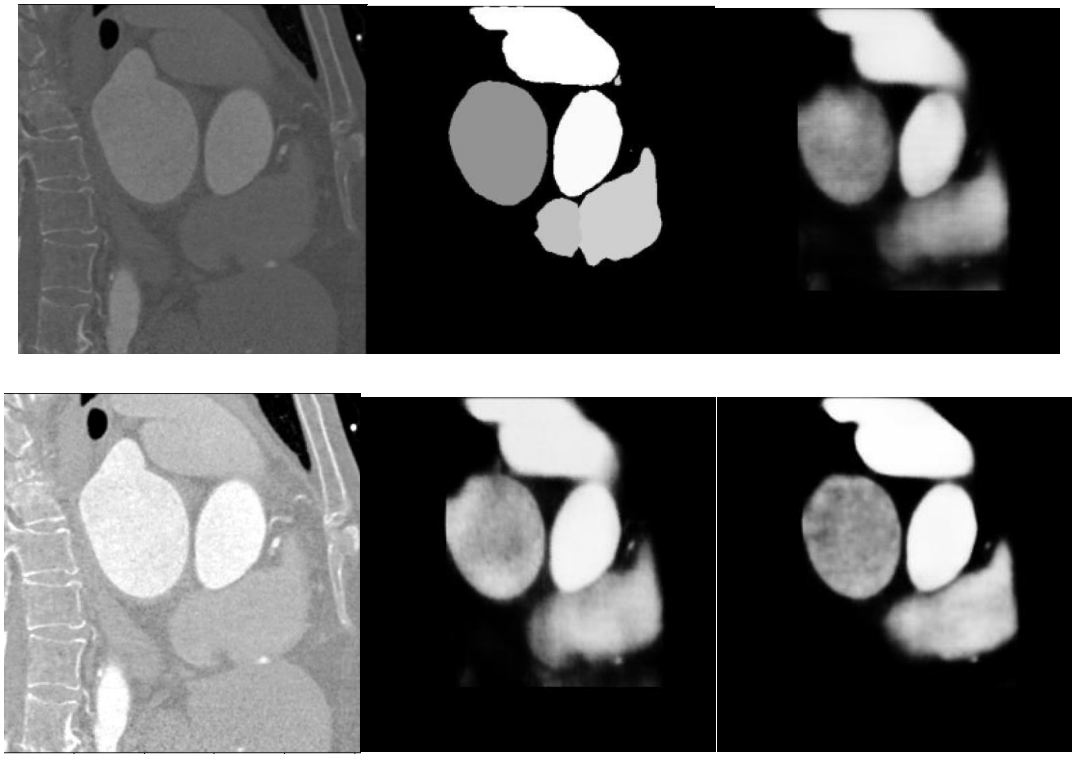
korištenjem CLAHE s većim clip limit-om, ali je rezultat lošiji u usporedbi s prethodnom metodom.

Sljedeća korištena metoda je sigmoid correction s parametrima cutoff = 0.35 i gain = 17 prikazana na slikama 5.21. - 5.26. Parametri su podešeni tako da odgovaraju najvećem broju slika. Ova metoda daje dobar kontrast uz minimalno dodavanje šuma za razliku od prethodno korištenih metoda gdje je šum bio izraženiji. Rezultat segmentacije je također bolji od originala te se vidi da je median filter u nekim područjima pomogao odrediti rubove, ali nije uklonio originalan šum sa slike zbog čega boja nije ujednačena koliko bi trebala biti. U usporedbi s prethodnim metodama daje ujednačeniju boju jer se na slici ne stvara šum te su rubovi približno slično definirani.



Sl. 5.21., 5.22., 5.23., 5.24 , 5.25. i 5.26. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa sigmoid correction (lijevo), rezultat sa sigmoid correction (sredina), sigmoid i median filter (desno) x presjeka

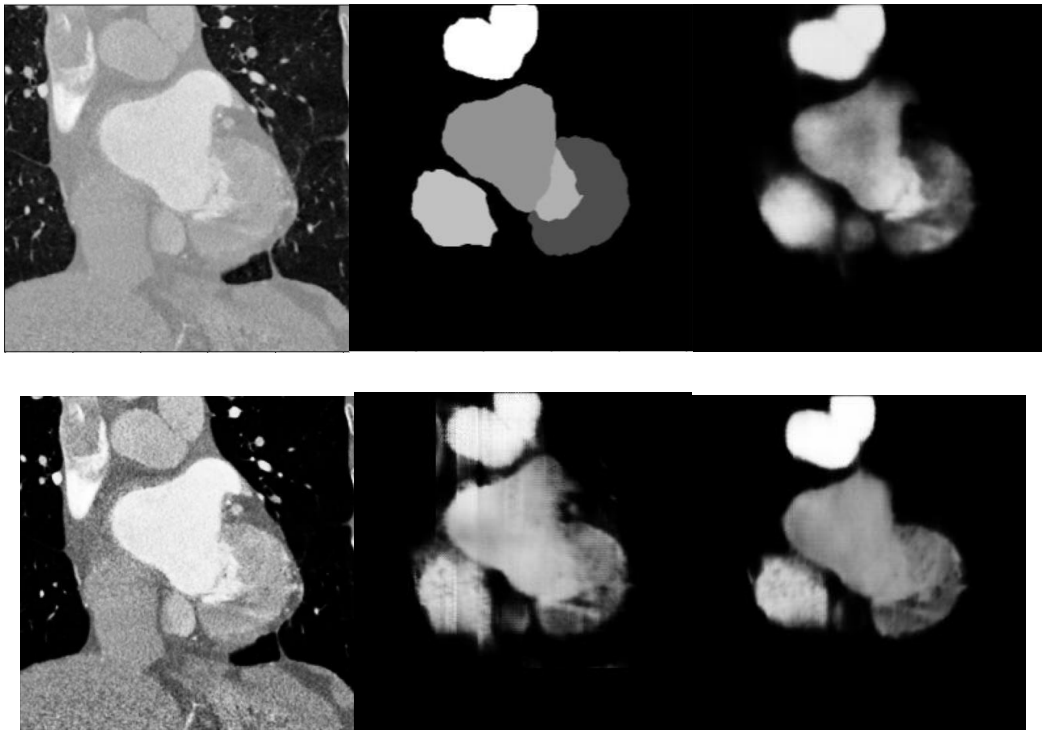
Na slikama 5.27. - 5.32 se vidi da je contrast stretching poboljšao predviđanje nekih dijelova te su rezultati cjelovitiji, ali u usporedbi s ostalim metodama ova je proizvela najlošije rezultate. Boja je većinski loše određena te u dijelovima i lošije od predviđanja bez predobrade (tamno sivi dio). Rubovi su bolje definirani i oblici su puniji te je rezultat bolji uz korištenje median filtera. Korištenje ove metode s rasponom [2%, 98%] vrijednosti piksela daje sliku koja najslabije djeluje na kontrast u usporedbi s prethodno spomenutima što je doprinijelo ovim rezultatima.



Sl. 5.27., 5.28., 5.29., 5.30., 5.31. i 5.32. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa contrast stretching (lijevo), rezultat sa contrast stretching (sredina), sa contrast stretching i median filterom (desno) x presjeka

Za x presjek sve metode su dale rezultate bolje od metode bez korištenja predobrade. Izjednačavanje histograma, CLAHE (clip limit = 0.03) i sigmoid correction uz korištenje median filtera.

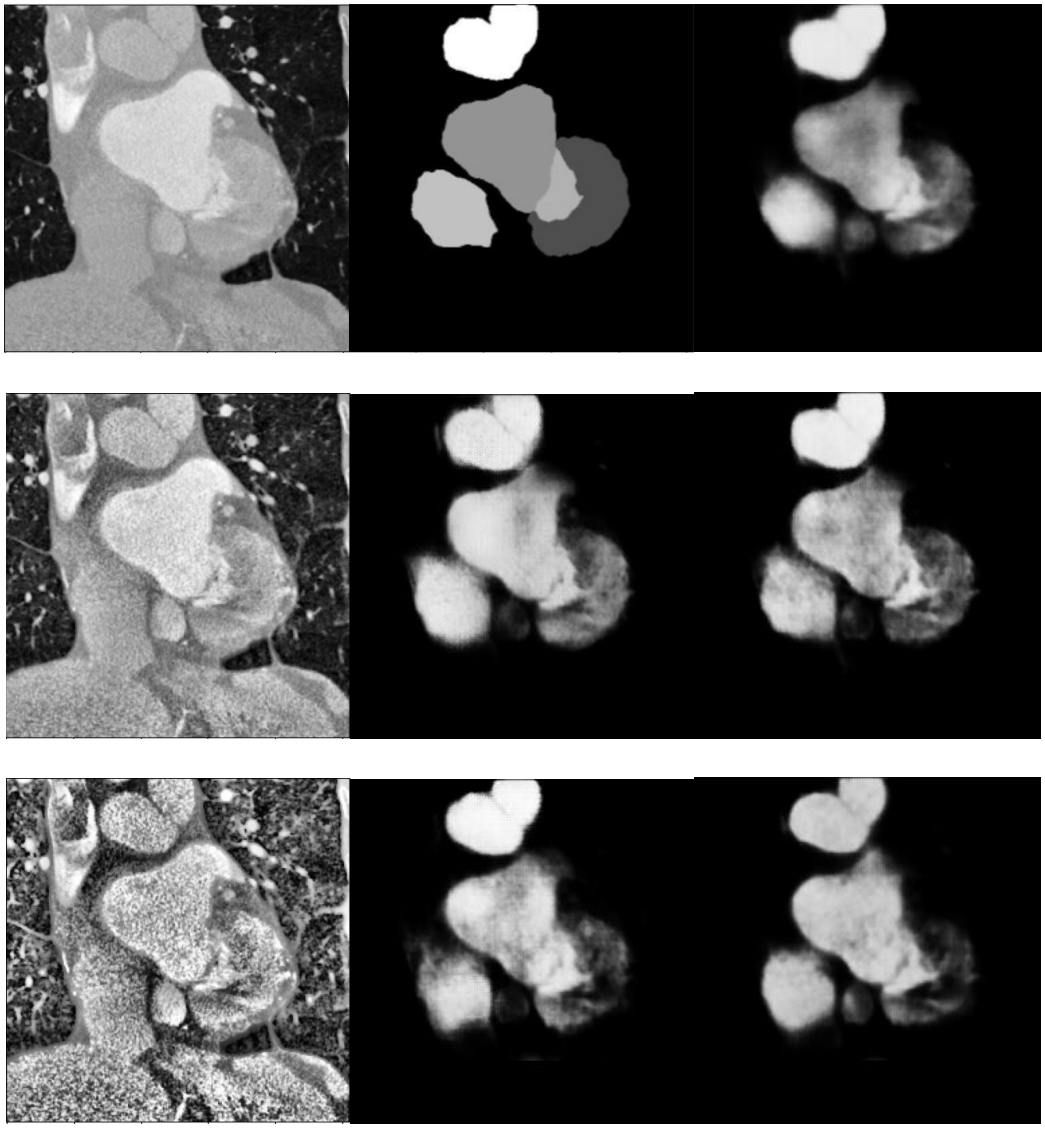
5.2.2. Y presjek



Sl. 5.33., 5.34., 5.35., 5.36., 5.37. i 5.38. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa izjednačavanjem histograma (lijevo), rezultat sa izjednačavanjem histograma (sredina), sa izjednačavanjem histograma i median filterom (desno) y presjeka

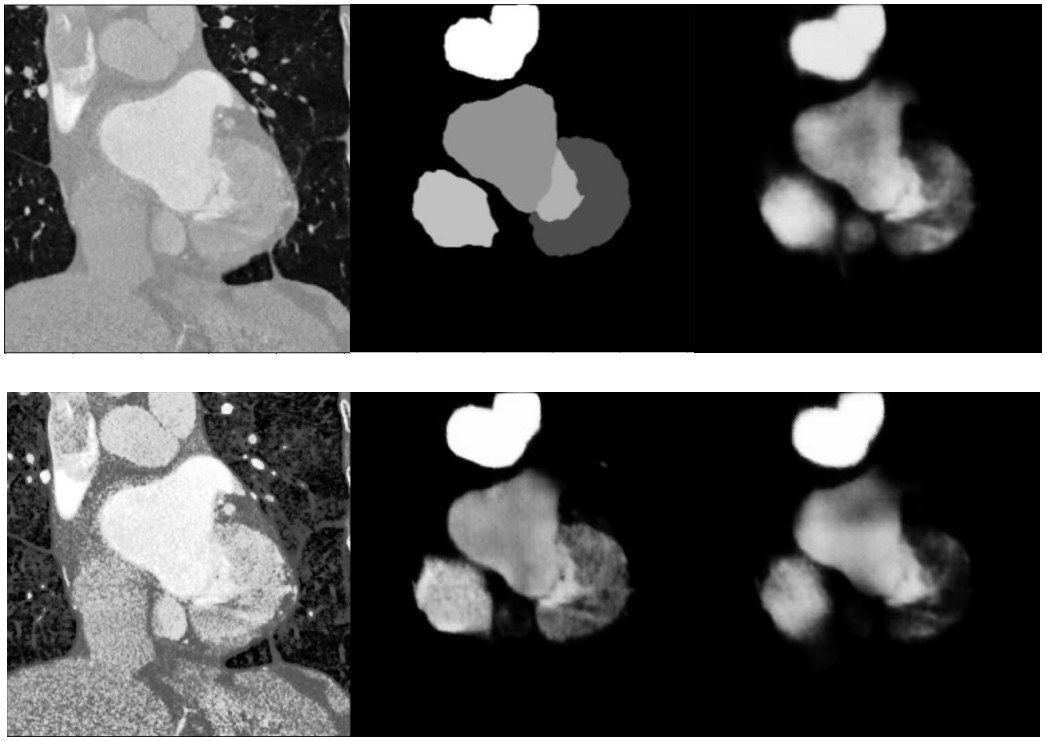
Korištenje izjednačavanja histograma u ovom slučaju nije dao dobre rezultate. Boja većinski nije pogođena, ali je ujednačenija. Rubovi su na nekim dijelovima bolje definirani, ali postoje smetnje na slici. Slika u nekim dijelovima djeluje bolje od originala, ali rezultat se nije poboljšao korištenjem ove metode. Korištenjem median filtera uklonjene su smetnje sa slike te je detekcija boja puno bolja i od originala i bez median filtera. Unatoč tome, vidljivo je da model nije uspio odrediti najtamniji dio koji je najviše neujednačen na cijeloj slici (najtamnije sivo). U slučaju ove metode konačan rezultat je bolji uz korištenje izjednačavanja histograma i median filtera, ali bez filtera rezultat nije dao dobre rezultate.

Na slikama 5.39. - 5.47 vidljivo je poboljšanje u detektiranju rubova u usporedbi s originalom te s izjednačavanje histograma. CLAHE (clip limit = 0.01) nije dobro odredio boje, ali su rubovi jasniji. Korištenjem median filtera boja je počela biti sličnija sivoj, ali je manje ujednačena nego prije te su rubovi slabije detektirani. Korištenje CLAHE (clip limit = 0.03) nije dalo puno poboljšanja te su rezultati u usporedbi s prethodnim lošiji. Korištenje median filtera pomoglo je da boja bude ujednačenija, ali je lošije pogođena. U dijelovima su se rubovi bolje detektirali uz median filter. U usporedbi s prethodnom metodom ni jedan rezultat nije jednako dobar kao izjednačavanje histograma uz median filter jer ni jedna metoda nije pogodila boju i rubove jednako dobro.



Sl. 5.39., 5.40., 5.41., 5.42., 5.43., 5.44., 5.45., 5.46. i 5.47. Prvi red: originalna slika (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa CLAHE clip limit = 0.01 (lijevo), rezultat sa CLAHE 0.01 (sredina), sa CLAHE 0.01 i median (desno), treći red: original sa CLAHE clip limit = 0.03

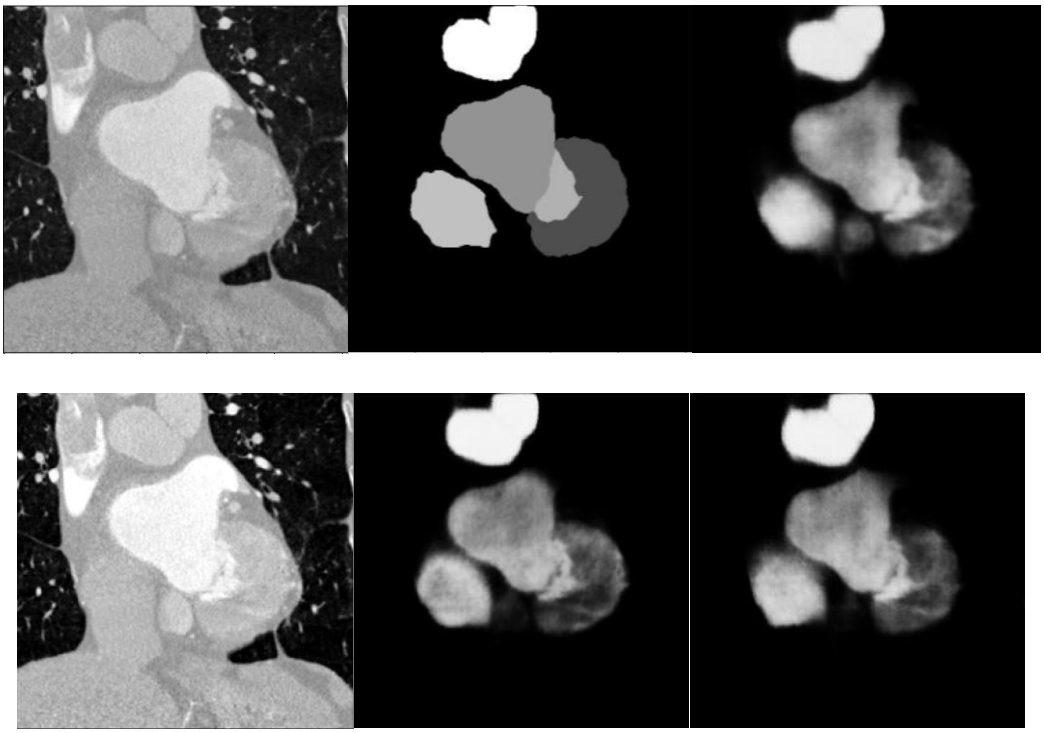
Na slikama 5.48. – 5.53. može se odmah zapaziti da sigmoid correction s parametrima cutoff = 0.6 i gain = 10 daje veliko poboljšanje. Boje su ujednačene te dobro detektirane i rubovi su dobro



Sl. 5.48., 5.49., 5.50., 5.51., 5.52. i 5.53. Prvi red: original (lijevo), očekivan rezultat (sredina) i rezultat bez predobrade (desno), drugi red: original sa sigmoid correction (lijevo), rezultat sa sigmoid correction (sredina), sa sigmoid correction i median filterom (lijevo) y presjeka

definirani. U ovom slučaju median filter je pogoršao rezultat. Ova metoda bez median filtera je dosad dala najbolje rezultate uz trenutne parametre.

Na slikama 5.54. – 5.59. vidi se da je contrast stretching dobro detektirao određene rubove te pogodio i boju iako ona nije potpuno ujednačena. Korištenje median filtera nema značajan utjecaj na rezultat u ovom slučaju.

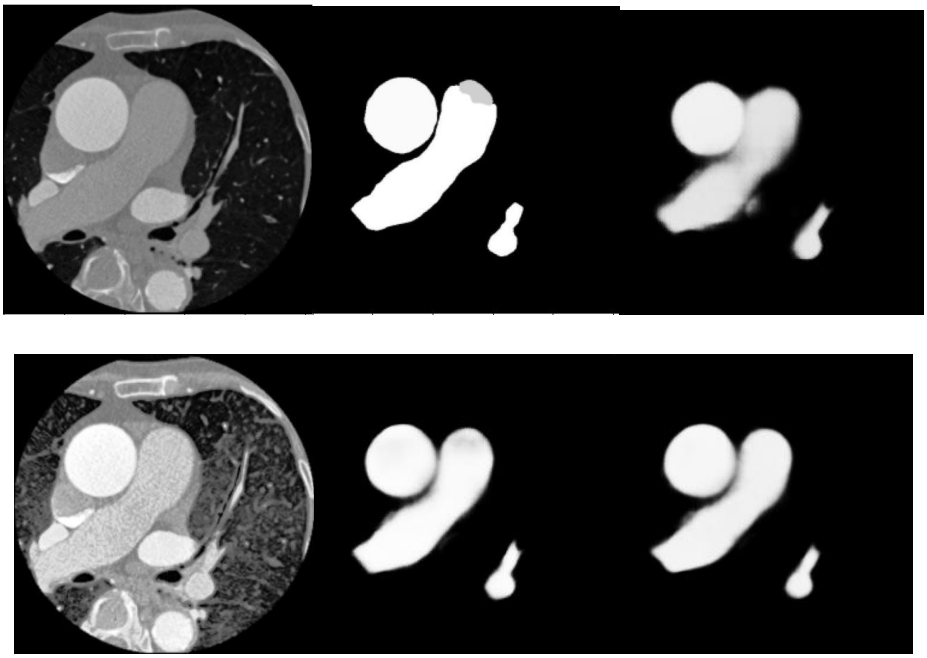


Sl. 5.54., 5.55., 5.56., 5.57., 5.58. i 5.59. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa contrast stretching (lijevo), rezultat sa contrast stretching (sredina), sa contrast stretching i median filterom (desno) y presjeka

Za y presjek najbolji rezultat proizveli su izjednačavanje histograma uz korištenje median filtera te sigmoid correction bez median filtera uz zadane parametre. Ostale metode nisu dale znatno poboljšanje u usporedbi s modelom bez predobrade. Iz priloženog se može vidjeti da metode koje daju određen rezultat za jedan presjek ne daju nužno isti za drukčiji jer su slike različitih presjeka različite u nekoliko bitnih područja. To su najčešće svjetlina slike, koliko jasno se vide određeni dijelovi potrebni za detekciju, koliko je velik kontrast između dijelova slike te koje boje prevladavaju u promatranim dijelovima slike. Zbog ovoga treba razmatrati korištenje različitih metoda za različit tip slike i različit presjek.

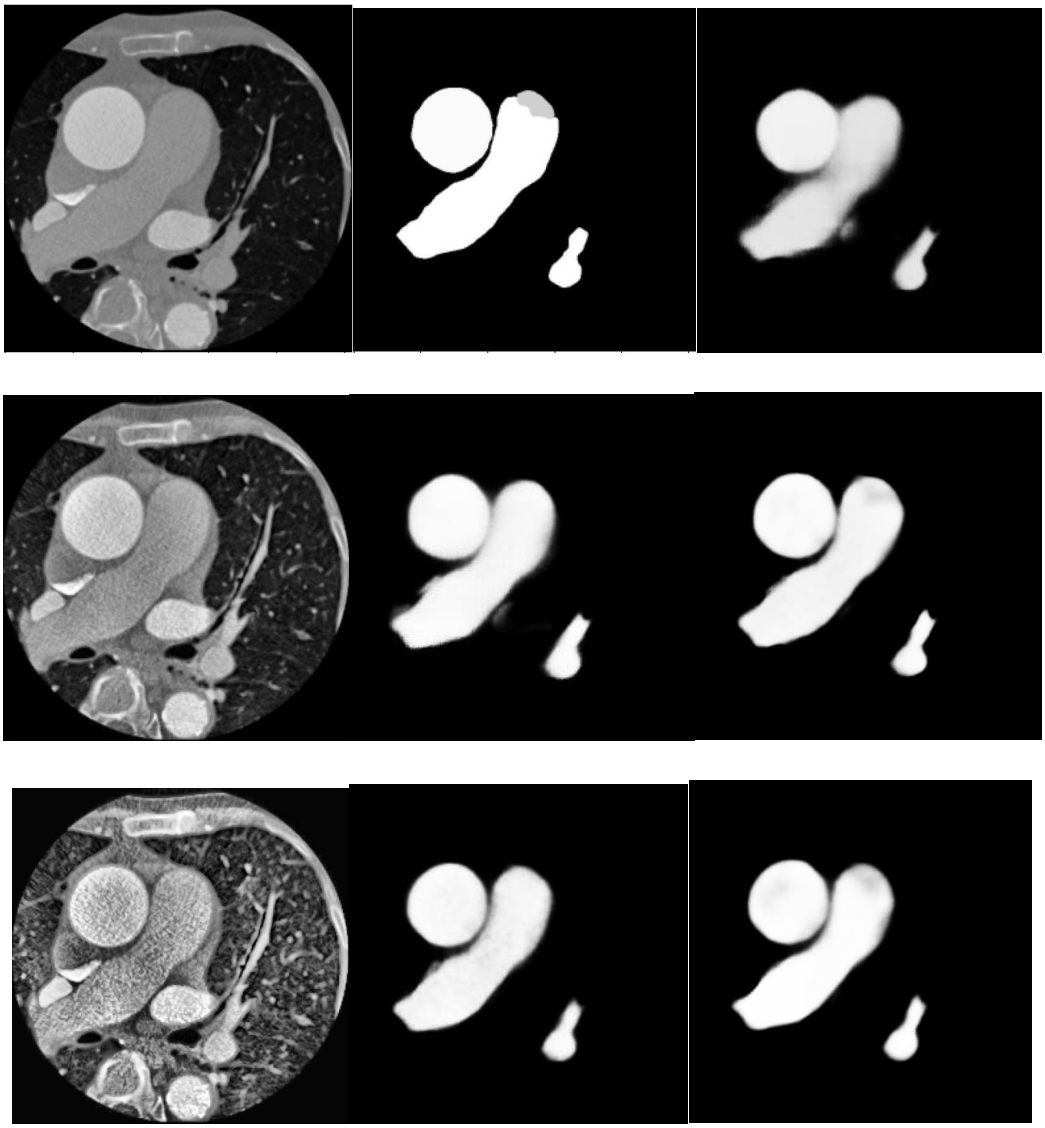
5.2.3. Z presjek

Na slikama 5.60. – 5.65. može se vidjeti da je izjednačavanje histograma poboljšao rezultat, posebno u području detektiranja rubova. Vidljive su i blage naznake da je detektirano malo tamnije sivo područje na slici. Primjenom i median filtera nema značajnih promjena.



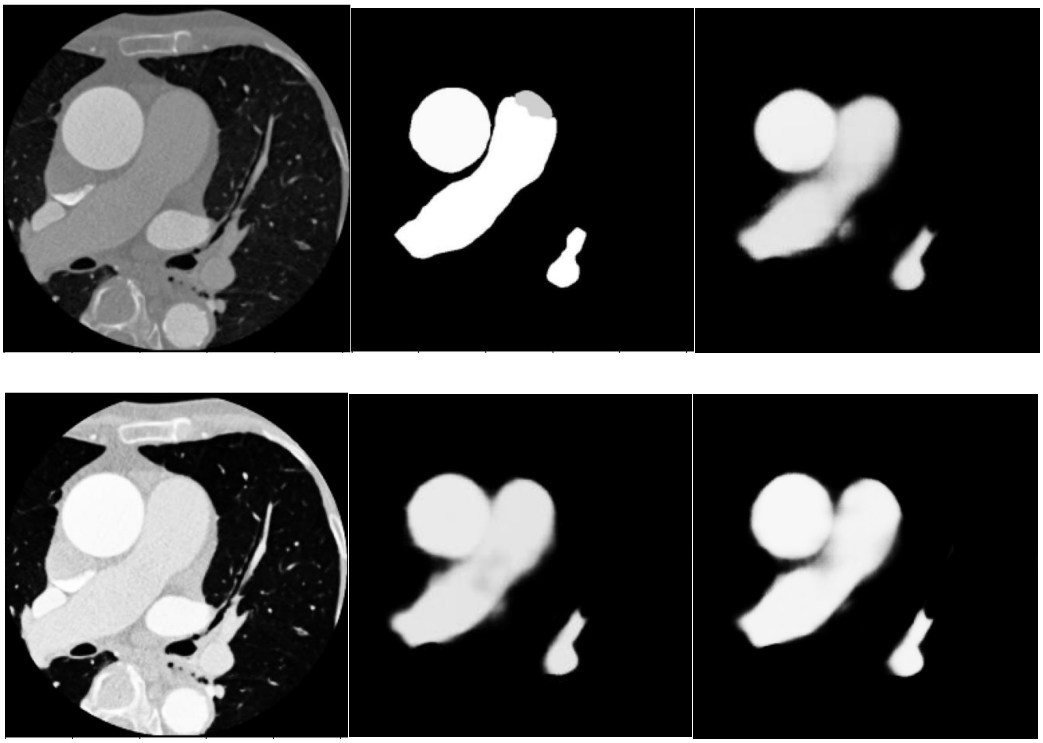
Sl. 5.60., 5.61., 5.62., 5.63., 5.64. i 5.65. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa izjednačavanjem histograma (lijevo), rezultat sa izjednačavanjem histograma (sredina), sa izjednačavanjem histograma i median filterom (desno) z presjeka

Na idućim primjerima na slikama 5.66. – 5.74. prikazano je korištenje metode CLAHE. Korištenjem CLAHE s parametrom clip limit = 0.01 blago je poboljšan rezultat. Na određenim mjestima rubovi nisu potpuno definirani slično kao u originalu, ali boja je punija u ovom slučaju. Korištenje median filtera ovdje je poboljšalo rezultat te se slabije vidi i sivi dio slike za razliku od slučaja ne korištenja median filtera. Povećanje clip limit parametra na 0.03 poboljšalo je detekciju rubova i dalo bolji rezultat, ali još uvijek nije detektiran sivi dio koji se tek javi u slučaju korištenja median filtera kao u prethodnom CLAHE primjeru. U slučaju korištenja CLAHE za z presjek korištenje median filtera je u oba slučaja pozitivno utjecalo na ishod.

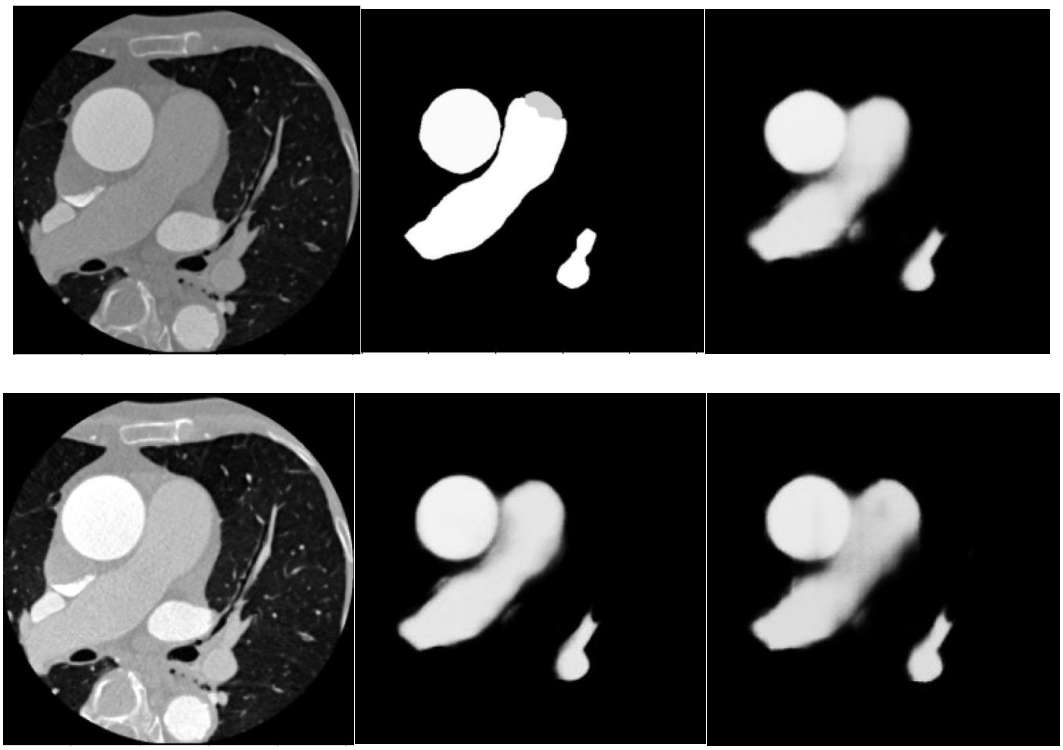


Sl. 5.66., 5.67., 5.68., 5.69., 5.70., 5.71., 5.72., 5.73. i 5.74. Prvi red: originalna slika (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa CLAHE clip limit = 0.01 (lijevo), rezultat sa CLAHE 0.01(sredina), sa CLAHE 0.01 i median (desno), treći red: original sa CLAHE clip limit = 0.03 (desno), rezultat sa CLAHE 0.03 (sredina), sa CLAHE 0.03 i median filterom (desno) z presjeka

U slučaju korištenja sigmoid correction s parametrima cutoff = 0.35 i gain = 10 nema velikog poboljšanja u usporedbi s modelom bez predobrade. Slika je punija, ali rubovi još uvijek nisu u potpunosti naglašeni. Korištenje median filtera u ovom slučaju nije pomoglo poboljšati rezultat. Ovdje bi bilo pogodno koristiti drukčije parametre za sigmoid correction, ali oni su bili podešeni za većinu slika, ali za ovu nisu bili najidealniji. Ova metoda iako je davala odlične rezultate u prethodnim primjerima ima nekonzistentnost jer izabrani parametri jako utječu na kvalitetu promjene kontrasta.



Sl. 5.81., 5.82., 5.83., 5.84 , 5.85. i 5.86. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa sigmoid correction (lijevo), rezultat sa sigmoid correction (sredina), sigmoid i median filter (desno) z presjeka



Sl. 5.75., 5.76., 5.77., 5.78 , 5.79. i 5.80. Prvi red: original (lijevo), očekivan rezultat (sredina), rezultat bez predobrade (desno), drugi red: original sa contrast stretching (lijevo), rezultat sa contrast stretching (sredina), sa contrast stretching i median filterom (desnoz presjeka

Na slikama 5.75. – 5.80. može se vidjeti zadnja korištena metoda, contrast stretching. Vidi se da metoda ne poboljšava rezultat znatno te korištenje median filtera u ovom slučaju samo pogoršava sliku.

Za z presjek najbolji rezultat dale su metode izjednačavanje histograma bez median filtera, CLAHE uz oba parametra i uz median filter. Iz ovih svih primjera može se vidjeti da metode velikom većinom pozitivno utječu na konačan rezultat te da su neke bolje od drugih ovisno o tipu slike i njenim svojstvima.

6. ZAKLJUČAK

U ovom radu razmatran je utjecaj različitih metoda predobrade u slučaju segmentacije CT slika srca korištenjem U-net neuronske mreže. Iz priloženih rezultata i primjera prikazano je da predobrada ima velikom većinom pozitivan utjecaj na konačan rezultat, ali razlike nisu uvijek velike ovisno o kakvoj se slici radi. Izjednačavanje histograma i CLAHE su se često pokazali najpouzdanijim uz korištenje median filtera. Sigmoid correction ostvario je odlične rezultate, uz veliku nekonzistentnost s obzirom na to da metoda jako ovisi o podešenim parametrima. Contrast stretching je većinom poboljšao sliku. Nije značajno utjecao na rezultat kao prethodne metode, ali je imao konzistentnost u rezultatima različitih presjeka. Median filter je velikim dijelom poboljšao rezultate ili nije utjecao na njih dok je na manjem broju slika ostvario pogrešan rezultat. Prema tome, može se zaključiti kako je za kvalitetnu predobradu slike potrebno pažljivo odabrati parametre. Konzistentnosti rezultata pridonijelo bi učiniti slike što sličnijima za različite presjeke tako da se slike ne razlikuju znatno u intenzitetu svjetlije. Ovo je još jedna predobrada koju bi bilo dobro obaviti i prije samog treniranja modela u sklopu pripreme za treniranje. To bi omogućilo jednostavniji odabir metode za poboljšanje slike te bolje rezultate.

SAŽETAK

Ovaj rad analizira metode predobrade pri uporabi za segmentaciju medicinskih slika srca dobivenih CT uređajem. Koristi metode promjene kontrasta izjednačavanje histograma, CLAHE, sigmoid correction i contrast stretching te metodu uklanjanja šuma median filter prije treniranja modela pomoću medicinskih slika pretvorenih iz 3D prostora (.nii) u 2D prostor (.jpg) kako bi model bio što točniji. Metoda strojnog učenja je U-net konvolucijska neuronska mreža koja ima veliku primjenu u segmentiranju medicinskih slika. Uz sve navedeno, rad pruža teorijsku osnovu svih korištenih metoda i tehnika te pojašnjava medicinsku pozadinu i uporabu slika. Na kraju rada provodi se analiza utjecaja spomenutih metoda predobrade te komentiraju rezultati.

Ključne riječi: CT, predobrada slika, U-net, segmentacija medicinckih slika, srce

ABSTRACT

This paper analizes preprocessing methods used in case of medical image segmentation of human heart taken by CT device. It uses contrast enhancement methods izjednačavanje histograma, CLAHE, sigmoid correction and contrast stretching, and uses noise removing median filter before training model using medical images converted from 3D space to 2D space so that model would have higher accuracy. Machine learning method used is U-net convolutional neural network which has big applications in medical image segmentation. With all listed, this paper provides theoretical background of all used methods and tehniques while explaining medical background and image usage. At the end effect of methods is analized and results are commented.

Keywords: CT, heart, image preprocessing, medical image segmentation, U-net

LITERATURA

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation",Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany 18 May 2015

[2] Hsin-Han Tsai, Chiou-Shann Fuh: "Liver Segmentation with 2D Unet", Department of Computer Science and Information Engineering, National Taiwan University, Taiwan

[3] Muhammed Islam, Kaleem, Nawaz Khan i Muhammed Salam Khan: "Evaluation of Preprocessing Techniques for U-net Based Automated Liver Segmentation", March 26, 2021.

[4] Marija Habjan, Irena Galić, Hrvoje Leventić i Danilo Babin: "Neural Netwoork Based Whole Heart Segmenttion from 3D CT Images", Volume 11, No 1, 2020.

[5] Akifumi Yoshida, Yongbum Lee, Norihiko Yoshimura, Tatsuya Kuramoto, Akira Hasegawa, Tsutomu Kanazawa: "Automated heart segmentation using U-Net in pediatric cardiac CT", Measurement:Sensors, Volume 18, December 2021.

[6] Sanguk Park and Minyoung Chung: "Cardiac Segmentation on CT Images through Shape-Aware Contour Attentions", 27 May, 2021.

[7] https://www.topdoctors.co.uk/medical-dictionary/ct-scan-cat

[8] https://www.fda.gov/radiation-emitting-products/medical-x-ray-imaging/computed-tomography-ct

[9] Brahim AIT SKOURT*, Abdelhamid EL HASSANI, Aicha MAJDA, „Lung CT Image Segmentation Using Deep Neural Networks“, Computer Science department, Faculty of Sciences and Technology of Fez, Morocco, Procedia Computer Science 127 (2018) 109–113

[10] Repozitorij za pretvobu formata alika: https://github.com/FNNDSC/med2image

[11] https://www.radiologycafe.com/frcr-physics-notes/ct-imaging/ct-image-quality/

[12] https://my.clevelandclinic.org/health/body/23074-heart-chambers

[13] https://www.bhf.org.uk/informationsupport/heart-matters-magazine/medical/tests/ct-scans-of-the-heart

[14] B. Kanchanadevi and P.R. Tamilselvi, „Preprocessing Using Image Filtering Method and Techniques for Medical Image Compression Techniques“, ICTACT Journal on Image and Video Processing, February 2020, Volume: 10, Issue: 03

[15] Aydin Demircioğlu, „The effect of preprocessing filters on predictive performance in radiomics“, European Radiology Experimental (2022) 6:40

[16] Kshema, Jayesh George Melekoodappattu, D.Anto Sahaya Dhas: „PREPROCESSING FILTERS FOR MAMMOGRAM IMAGES: A REVIEW“, Proc. IEEE Conference on Emerging Devices and Smart Systems (ICEDSS 2017) 3-4 March 2017, Mahendra Engineering College, Tamilnadu, India

[17] Keumsun Park, Minah Chae, and Jae Hyuk Cho: „Image Pre-Processing Method of Machine Learning for Edge Detection with Image Signal Processor Enhancement“. Objavljeno online 11.1.2021.

[18] https://pyimagesearch.com/2022/02/21/u-net-image-segmentation-in-keras/

[19] https://www.analyticsvidhya.com/blog/2022/10/image-segmentation-with-u-net/

[20] S.Perumal and T.Velmurugan: „Preprocessing by Contrast Enhancement Techniques for Medical Images“, International Journal of Pure and Applied Mathematics, Volume 118 No. 18 2018, 3681-3688

[21] Kim-Ngan NGUYEN-THI1, Ha CHE-NGOC2, Anh-Thy PHAM-CHAU „An Efficient Image Contrast Enhancement Method using Sigmoid Function and Differential Evolution“, Journal of Advanced Engineering and Computation (JAEC), Volume: 4, Issue: 3, 2020, September

[22] https://medium.com/geekculture/semantic-image-segmentation-using-unet-28dbc247d63e

[23] Naglaa Hassan, Norio Akamatsu: "A New Approach for Contrast Enhancement Using SIgmoid Function", The Internacional Arab Journal of Information Technology, Vol. 1, No.2, July 2004.

[24] Hari Babu Srivastava , Vinod Kumar , H.K. Verma , and S.S. Sundaram: „Image Pre-processing Algorithms for Detection of Small/Point Airborne Targets“, Defence Science Journal, Vol. 59, No. 2, March 2009, pp. 166-174

[25] https://datagen.tech/guides/image-annotation/image-segmentation/

[26] https://scikit-image.org/

[27] https://docs.aspose.com/ocr/net/image-processing/

[28] https://scikit-image.org/skimage-tutorials/lectures/4_segmentation.html

[29] https://theailearner.com/2019/01/30/contrast-stretching/

ŽIVOTOPIS

Daria Ćaćić rođena je u 11. siječnja 1998. u Osijeku. Prvu godinu osnovne škole završila je u Osnovnoj školi Bratoljuba Klaića Bizovac i ostatak u Osnovnoj školi Josipovac nakon selidbe. Završila je osnovnu školu te upisala Isusovačku klasičnu gimnaziju u Osijeku.

Nakon završetka srednje škole upisuje sveučilišni preddiplomski studij Računarstva na fakultetu Elektrotehnike, računarstva i informacijskih tehnologija u Osijeku te nakon toga upisuje diplomski studij na smjeru Informacijske i podatkovne znanosti (DRD). Zadnjih godinu dana uz fakultativne obaveze radi kao inženjer software-a u tvrtki ORQA u Osijeku radeći na projektima za dronove i vojne kamere.

_____

Potpis autora