# Generiranje sintetičkih projekcija u jednofotonskoj emisijskoj tomografiji koristeći duboke konvolucijske neuralne mreže

**Simić, Srđan Daniel**

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

**Srđan Daniel Simić**

# GENERIRANJE SINTETIČKIH PROJEKCIJA U JEDNOFOTONSKOJ EMISIJSKOJ TOMOGRAFIJI KORISTEĆI DUBOKE KONVOLUCIJSKE NEURONSKE MREŽE

DIPLOMSKI RAD

Pula, svibanj, 2021. godine

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

**Srđan Daniel Simić**

# GENERIRANJE SINTETIČKIH PROJEKCIJA U JEDNOFOTONSKOJ EMISIJSKOJ TOMOGRAFIJI KORISTEĆI DUBOKE KONVOLUCIJSKE NEURONSKE MREŽE

DIPLOMSKI RAD

**JMBAG: 0303056019, redoviti student**
**Studijski smjer: Informatika**

**Kolegij: Izrada informatičkih projekata**
**Znanstveno područje : Društvene znanosti**
**Znanstveno polje : Informacijske i komunikacijske znanosti**
**Znanstvena grana : Informacijski sustavi i informatologija**

**Mentor: doc. dr. sc. Nikola Tanković**

Pula, svibanj, 2021. godine

**Sažetak**

U zdravstvu učestale su dugačke liste čekanja za različite predglede i snimanja o kojima može ovisiti zdravlje i dobrobit pacijenta. Razlog takvim listama često se objašnjava nedostatkom opreme i radne snage u tom sektoru. Dok se ne poveća kapacitet radne snage i nadomjesti nedostajuća oprema, potrebno je tražiti riješenja za taj problem koristeći nove tehnologije. Neuronske mreže i tehnike dubokog učenja danas imaju široku primjenu u raznim područjima znanosti, zahvaljući velikom broju podataka te snazi modernih grafičkih kartica, pa tako i u medicini. U ovome radu želimo prikazati rješenje temeljeno upravo na toj tehnologiji koje bi doprinjelo smanjenju listi čekanja za snimanja u jednofotonskoj emisijskoj tomografiji.

Rad je podjeljen u nekoliko poglavlja gdje ćemo prvo predstaviti tehnologije koje će se koristiti, koje su neuronske mreže i jednofotonska emisijska tomografija, te ćemo nakon toga dati pregled implementacije rješenja za generiranje sintetičkih projekcija i zaključak.

**Ključne riječi** : duboke konvolucijske neuronske mreže, jednofotonska emisijska tomografija, generiranje sintetičkih projekecija

**Abstract**

In healthcare, long waiting lists for various examinations and scans on which the health and the well-being of the patient may depend have become frequent. The reason for such waiting lists is often explained by the lack of equipment and workforce in the sector. Until the workforce capacity is increased and the missing equipment is acquired, solutions to this problem need to be sought using new technologies. Neural networks and deep learning techniques are widely used today in various fields of science, thanks to the large amount of data and the power of modern graphics cards, including medicine. In this paper, we want to present a solution based on this technology that would contribute to reducing the waiting list for scans in single-photon emission computed tomography.

The paper is divided into several chapters where we will first present the technologies to be used which are neural networks and single photon emission computed tomography and after that we will give an overview of the implementation of solutions for generating synthetic projections and a conclusion.

**Keywords** : deep convolutional neural networks, single-photon emission computed tomography, generating synthetic projections

# Sadržaj

# 1 Uvod

Duboko učenje i duboke neuronske mreže pokazale su se učinkovitima u raznim grana medicine pa tako i u nuklearnoj medicini i jednofotonskoj emisijskoj tomografiji. Koristile su se u razne svrhe neke od kojih su smanjenje šuma u slika [1, 2], poboljšanja kvalitete slika uzrokovanih smanjenjem doze radiofarmaka [3, 4, 5, 6], dijagnostika bolesti [7, 8, 9], izrade rekonstrukcijskih algoritma za jednofotonsku emisijsku tomografiju [10, 11] i korekciju atenuacije [12, 13, 14].

Ovim radom želimo predstaviti rješenje za problem dugih lista čekanja za snimanja u jednofotonskoj emisijskoj tomografiji uzrokovanih nedostatkom radne snage kao i potrebne opreme. Točnije želimo predstaviti rješenje kojim bi se skratio proces snimanja pacijenata na pola, ne narušavajući pritom kvalitetu snimki, što bi za rezultat imalo povećanje broja pacijenta nad kojima se samo snimanje može izvršiti. Skraćivanje procesa snimanja postigli bi tako što bi smanjili broj kuteva pod kojim se pacijent snimi. No samo smanjivanje broja kuteva nije dovoljno jer time se postižu snimke niže kvalitete. Kako ne bi gubili na kvaliteti snimke zbog smanjenja broja kuteva, predlažemo naše rješenje, temeljeno na tehnologiji dubokih neuronskih mreža, pomoću kojeg možemo generirati sintetičke projekcije koje bi nadomjestile nedostajuće projekcije uzrokovane skraćenim vremenom snimanja. Važno je za napomenut da ovo nije potpuno novi pristup te da je generiranje nedostajućih projekcija rađeno i u [15].

U drugom poglavlju prvo će se opisati kratka povijest neuronskih mreža te njihova moderna arhitektura i rad, prikazat će se pozicija dubokog učenja unutar znanstvenih grana i prednosti s obzirom na klasično strojno učenje te će se navesti neke od najčešćih arhitektura modela dubokog učenja, objasnit će se konvolucijski sloj što je važna gradivna jedinica našeg rješenja i na kraju će biti opisane aktivacijske funkcije koje su se koristile prilikom izrade rješenja.

Treće poglavlje daje pregled o jednofotonskoj emisijskoj tomografiji te je opisan proces snimanja kao i razlozi zbog kojih se snimanja obavljaju, predstavljena je kratka povijest nuklearna medicine te njen opis i za kraj objašnjena je vrsta fantom čije su se projekcije koristile prilikom izrade rješenja.

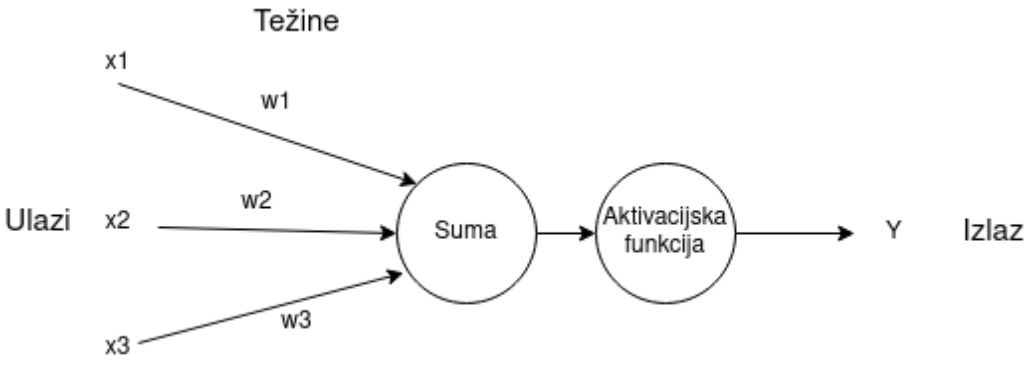
U četvrtom poglavlju predstavljamo implementaciju našeg rješenje. Prvo će se obrazložiti uloga rješenja u procesu snimanja jednofotonske emisijske tomografije te će se prikazati slučajevi upotrebe. Potom slijedi opis seta podatka koji je korišten kao i način na koji se isti pripremio za potrebe modela. Bit će objašnjeni modeli koji će se koristiti kao i proces treniranja istih. Nakon toga opisat ćemo proces testiranja kvalitete sintetičkih projekcija koji

je obavljen. Na kraju poglavlja predstavljamo proces generiranja potpunog seta projekcija čime zaokružujemo pregled implementacije rješenja.

U petom poglavlju dajemo zaključak o provedenom istraživanju u kojem smo naveli nedostatke kao i moguće smjerove za buduća istraživanja.

# 2 Neuronske mreže

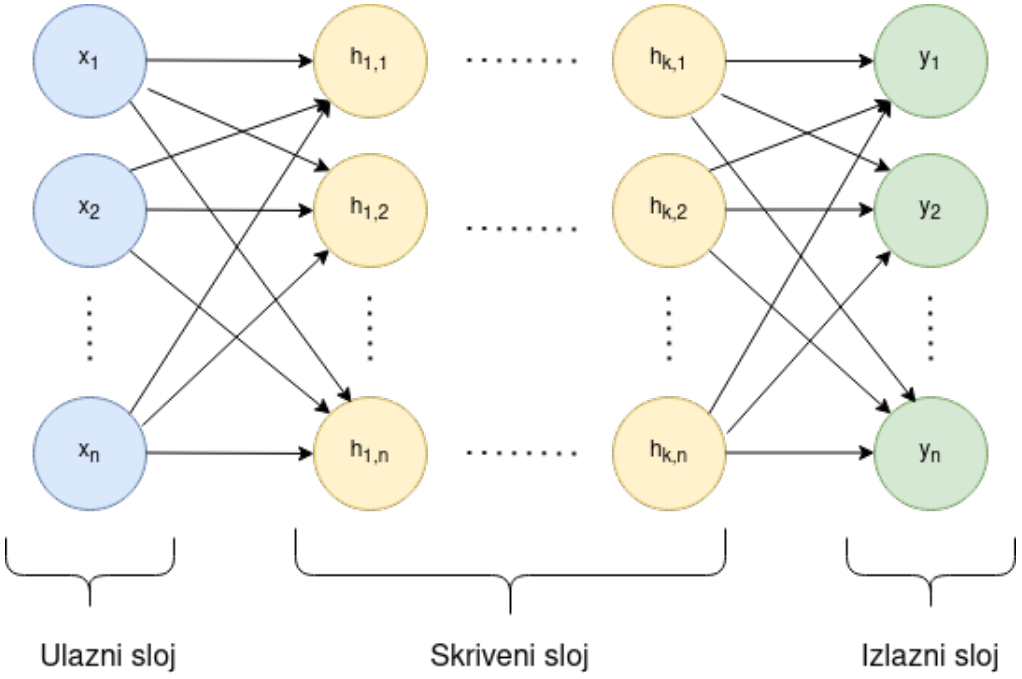
U [16], 1943. godine, prvi je put prezentiran matematički model biološkog neurona te time započinje era neuronskih mreža. Istraživanja su se nastavljala te se 1958. godine u [17] objavljuje perceptron. Sastojao se od ulaznog sloja, težina, sume ulaznih vrijednosti množene s težinama i aktivacijske funkcije, što možemo vidjeti iz slike 1, te se mogao se koristiti za binarnu klasifikaciju. No perceptron je imao nedostatke od kojih je jedan prezentiran u [18] te je bio da perceptron ne može naučiti xor funkciju. Za rješenje tog problema kreiran je više slojni perceptron (*eng. Multilayer Perceptron*) što je zapravo slaganje višestrukih perceptrona. Progres razvoja se tu usporio, zbog problema treniranja više slojnog perceptrona, sve do 1986. godine i rada [19] koji je prezentirao novu proceduru učenja *backpropagation* koja se koristi i danas. Ta procedura može se sažeti tako što za svaku instancu treninga, *backpropagation* algoritam prvo radi predikcije i mjeri grešku, onda prolazi kroz svaki sloj unazad i mjeri doprinos svake veze grešci te na kraju podešava težine veze kako bi se smanjila pogreška [20]. U sljedećem periodu 90-ih godina 20. stoljeća ponovo opada zanimanje za neuronske mreže, a tom je uzrok razvoj metoda potpornih vektora (*eng. Support Vector Machine* koji su davali bolje rezultate nego neuronske mreže. Posljednji uspon neuronskih mreža započinje 2006. godine radom [21] koji je jedan od mnogih koji potiču ponovni uspon u korištenju. Tada se događa rebrendiranje istraživanja neuronskih mreža u duboko učenje te do danas ostaje popularno područje istraživanja s učestalim objavama istraživačkih radova.




Slika 1: Arhitektura perceptrona

Današnje neuronske mreže kompleksne su i imaju mnogo slojeva i čvorova, osim ulaznog i izlaznog sloja sastoji se od mnogo takozvanih skrivenih slojeva koji se nalaze između njih kao što možemo vidjeti iz slike 2. Tih skrivenih

slojeva može biti od jedan do $k$ te ako ih ima više od tri takva mreža se smatra algoritmom dubokog učenja.




Slika 2: Prikaz moderne arhitekture neuronskih mreža

Rad neuronske mreže može se opisati tako da svaki čvor promatramo kao perceptron što znači za dobivanje određene vrijednosti iz čvora potrebno je napraviti sumu svih ulaza u taj čvor množen s njihovim pripadajućim težinama. Nakon toga pomoću aktivacijske funkcije, kojoj se predaje prethodna suma, određuje se izlaz čvora. Takva procedura ponavlja se za svaki čvor u svim slojevima te se dobiva rezultat u izlaznom čvoru. Funkcijom gubitka izračuna se greška i gradijent te se pomoću *backpropagation* algoritma, opisanog ranije, podešavaju težine kako bi se smanjila greška.

Kao i svaki model strojnog učenja i neuronske mreže potrebno je trenirati da postignemo željene rezultate te u području treniranja razlikujemo sljedeće tri metode :

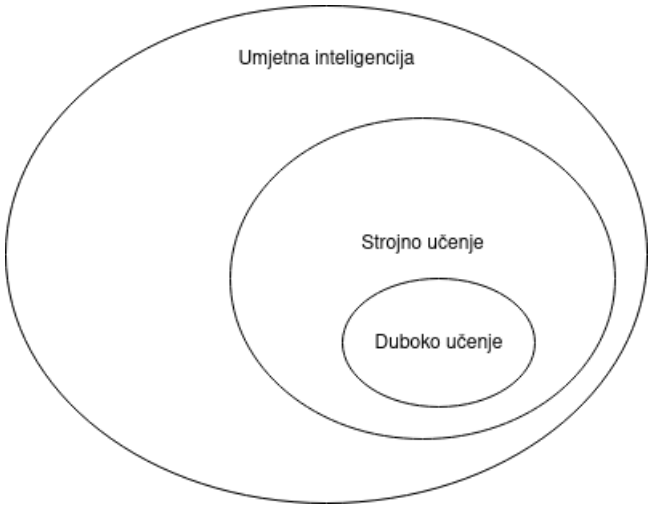
- Nadgledano učenje gdje se koriste označeni podaci
- Nenadgledano učenje gdje se uče uzorci u podacima koji nisu označeni
- Polu nadgledano učenje gdje se kombinira mali broj označenih podataka s velikim brojem ne označenih

Neuronske mreže koriste se u raznim problemima strojnog učenje, a posebno dobre rezultate pokazuju u sljedećim područjima :

- Računalni vid
- Prepoznavanje govora
- Procesiranje prirodnog jezika
- Bioinformatika
- Dizajniranje ljekova
- Robotika

## 2.1 Duboko učenje

Neuronske mreže građevna su jedinica dubokog učenja. Poziciju dubokog učenja unutar znanstvenih grana možemo vidjeti iz slike 3 te možemo primijetiti da je duboko učenje podskup strojnog učenja, a strojno učenje podskup umjetne inteligencije. Od klasičnog strojnog učenja, algoritmi dubokog učenje razlikuje se upravo po tome što se koriste neuronske mreže s mnogo skrivenih slojeva. Još jedna značajna razlika između klasičnog strojnog učenja i dubokog učenja je u tome što kod klasičnog strojnog učenja potrebno je relevantne značajke podatka ručno odrediti te se onda na temelju tih značajki kreira model za razliku od dubokog učenja gdje se korak određivanja relevantnih značajki odvija automatski direktno iz podataka. Prednost modela dubokog učenja jest da se oni često nastavljaju poboljšavati kako se veličina podataka povećava za razliku od modela klasičnog strojnog učenja koji dostižu vrhunac performansi s određenim brojem podataka.




Slika 3: Prikaz dubokog učenja unutar umjetne inteligencije

Postoje razne vrste arhitektura modela dubokog učenja koji se koriste u raznim istraživačkim područjima no neki od najčešće korištenih su sljedeći :

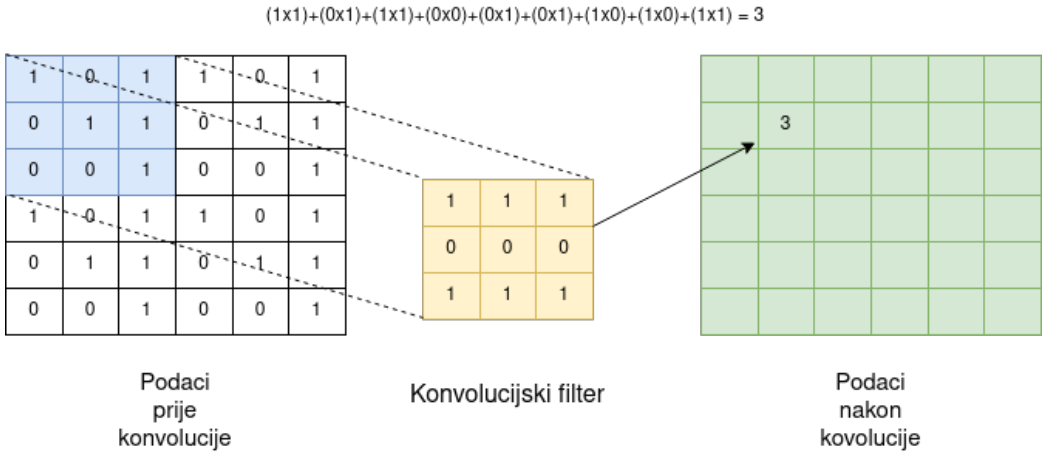
- Generativne suparničke mreže (*eng. Generative Adversarial Networks, GAN*). Objavljene u radu [22]. GAN se sastoji od dvije duboke mreže koje se nazivaju generator i diskriminator. Generator uči generirati nove setove podataka na temelju trening seta podataka dok diskriminator pokušava razlikovati generirane podatke od onih iz trening seta.
- Konvolucijske neuronske mreže (*eng. Convolutional Neural Networks, CNN*). Prva uspješna CNN arhitektura opisana je u radu [23]. Kod

klasičnih CNN arhitektura osnovni gradivni elementi su konvolucijski slojevi, slojevi za udruživanje(*eng. pooling layers*) i potpuno povezani sloj(*fully connected layer*).

- Ponavljajuće neuronske mreže (*eng. Recurrent Neural Networks, RNN*). Prvi puta se spominju u [24]. Njihova posebnost je u tome što prilikom obrade trenutnih ulaznih podataka u obzir uzimaju i prethodne ulazne podatke te se može reći da izlazni podaci iz takvih mreža ujedno postaju i ulazni.

## 2.2 Konvolucijski sloj

Konvolucijski slojevi koriste se u izradi konvolucijski neuronskih mreža koje se koriste za procesiranje podataka koje imaju rešetkastu topologiju. Primjer takvih podataka jesu dvodimenzionalne slike. Glavna značajka konvolucijskog sloja je konvolucija. Nju možemo definirati za dvodimenzionalne slike kao skalarni produkt matrice vrijednosti, *kernel*, sa svakim susjedstvom u ulazu [25]. Za *kernel* možemo reći da je konvolucijski filter , a susjedstvo u ulazu bilo bi određeno područje ulaza na kojem se konvolucijski filter nalazi. Bitno je naglasiti da vrijednosti konvolucijskog filter su inicijalizirane nasumično te se ažuriraju pomoću *backpropagation* algoritma. Izlazna slika stvara se tako što se konvolucijski filter pomiče po svim područjima ulaza te ponavlja operaciju skalarnog produkta. Jedan korak u tom procesu konvolucije prikazan je slikom 4. Također kao posljedicu konvolucije možemo primijetiti da su podaci, odnosno dvodimenzionalna slika, drugačijeg oblika nakon konvolucije.




Slika 4: Rad konvolucije unutar konvolucijskog sloja

Ako uzmemo za primjer rad PyTorch konvolucijski sloj možemo primijetiti koji još parametri utječu na promjenu oblika. Promjene nad podacima računaju se po sljedećim formulama :

$$H_{izlaz} = \frac{H_{ulaz} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1$$

$$W_{izlaz} = \frac{W_{ulaz} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1$$

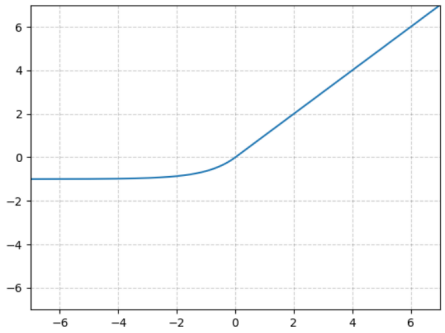
Prvo možemo reći da $H$ i $W$ označavaju visinu i širinu podataka, odnosno dvodimenzionalne slike. Dalje slijedi objašnjenje sljedećih parametara:

- *Padding*. Određuje koliko ćemo dodatnih takozvanih *ghost pixels*, koji imaju vrijednost nula, dodati sa svake strane ulaza.
- *Dilation*. Određuje razmak između elementa konvolucijskog filtera.
- *Kernel_size*. Određuje veličinu našega konvolucijskog filtera.
- *Stride*. Određuje korak za unakrsnu korelaciju, odnosno za klizeći skalarni produkt.

Možemo primijetiti da kombinacijom ovih parametara možemo na izlazu dobiti podatke istoga oblika ali i manjeg ili većeg.

## 2.3 Aktivacijske funkcije

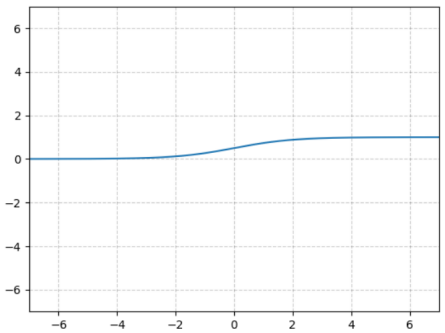
Aktivacijske funkcije služe za određivanje vrijednosti na izlazu čvorova neuronskih mreža te za određivanje hoće li se čvor aktivirati ili ne. Ovisno o tipu aktivacijske funkcije i vrijednosti na njenom ulazu dobijamo različite vrijednosti na izlazu. Prikazat ćemo aktivacijske funkcije koje su se koristile pri izradi ovog rada.




Slika 5: Prikaz ELU aktivacijske funkcije, na $x$ osi su vrijednosti ulaza a na $y$ osi vrijednosti izlaza

Eksponencijalna linearna jedinica (*eng. Exponential Linear Unit, ELU*) varijacija je Ispravljene linearne jedinice (*eng. Rectified Linear Unit, ReLU*) u kojoj je modificiran nagib na negativnom dijelu funkcije, što možemo vidjeti iz slike 5, te se po tome i razlikuje od ReLU koji za sve negativne vrijednosti daje rezultat 0. Formula po kojoj radi ELU aktivacijska funkcija je sljedeća:

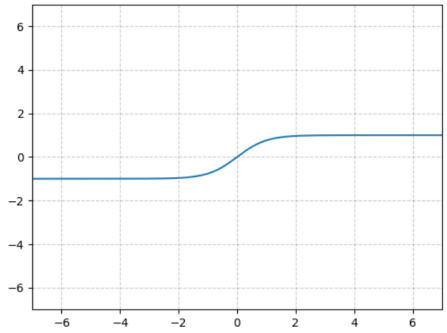
$$ELU(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (e^x - 1), & \text{if } x \leq 0 \end{cases}$$




Slika 6: Prikaz Sigmoid aktivacijske funkcije, na $x$ osi su vrijednosti ulaza a na $y$ osi vrijednosti izlaza

Sigmoidna (*eng. Sigmoid*) aktivacijska funkcija ulazne vrijednosti pretvara u vrijednosti u rasponu od 0 do 1 što se može uvidjeti iz slike 6. Formula po kojoj radi Sigmoid je sljedeća:

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$




Slika 7: Prikaz Tanh aktivacijske funckije, na $x$ osi su vrijednosti ulaza a na $y$ osi vrijednosti izlaza

Hiperbolični tangen (*eng. Hyperbolic tangen, Tanh*) aktivacijska funkcija veoma je slična Sigmoid aktivacijskoj funkciji no za razliku od nje Tanh pretvara ulazne vrijednosti u raspon od -1 do 1, kao što je vidljivo iz slike 7. Formula po kojoj radi Tanh je sljedeća :

$$\text{Tanh}(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

# 3 Jednofotonska emisijska tomografija

Jednofotonska emisijska tomografija(*eng. single photon emission computed tomography, SPECT*) medicinska je tehnika razdiobe radioaktivnosti u tijelu pacijenta koja se bazira na metodama konvencionalne nuklearne medicine i tomografske rekonstrukcije te pruža informacije na temelju prostorne koncentracije ubrizganog radiofarmaka. Uređaj koji se koristi tijekom snimanja prikazan je slikom 8 te se sastoji od gama kamere koja se rotira i snima gama zrake emitirane iz tijela pacijenta koje nastaju zbog radioaktivnog raspada radiofarmaka.

Slika 8: Prikaz SPECT uređaja. Preuzeto s [26]

Tijekom snimanja i rotacije gama kamere, gdje je klinička praksa da u slučaju dvoglave kamere svaka kamera napravi 180 stupnjeva rotacije, kreiraju se 2D slike, projekcije, iz različitih kuteva. Veličina takvih projekcija u praksi je obično matrica 64x64 ili 128x128. Broj kuteva pod kojim će se slika kreirati prethodno je određena na uređaju, a praksa je da se namjesti na svakih tri do šest stupnjeva. Vrijeme potrebno za snimiti svaku projekciju je varijabilno no u prosjeku traje manje od minute. Nakon što proces snimanja završi započinje se s rekonstrukcijom snimke u 3D model pomoću specijaliziranih rekonstrukcijski algoritama. Takav 3D model omogućuje liječniku da dobije uvid u presjek organ po svakoj željenoj osi. Za razliku od drugih

načina snimanja u medicini koji pružaju uvid u izgled organa, SPECT služi za prikazivanje rada organa. Navest ćemo neke od razloga zašto se obavlja SPECT snimanje :

- Dijagnostika bolesti mozga
- Analiza protoka krvi kroz mozak, srce i pluća
- Pregled kostiju, upalni procesi i metastatske bolesti
- Dijagnostika bolesti bubrežnog sustava
- Dijagnostika onkoloških bolesti

## 3.1 Nuklearna medicina

Početci nuklearne medicine mogu se naći u zadnjim godina devetnaestog stoljeća s otkrivanjem radioaktivnosti i radiuma. Biološki temelji nuklearne medicine su napravljeni između 1910. i 1945. tijekom kojih se razvijali radiofarmaci za primjenu u živim bićima. Razvoj tehnologije 1950-ih omogućio je dobivanje slika distribucije radiofarmaka u tijelu pacijenta. Glavne prekretnice u tom razdoblju bile su razvoj pravocrtnog skenera 1951. od strane Benedikt Cassena i Anger kamere, preteče svim modernim jednofotonskim sistemima za snimanje, 1958. od strane Hal Angera, prikazane slikom 9. Tijekom 1970-ih godina razvijena je matematika za rekonstrukciju tomografskih slika iz seta kutnih slika oko pacijenta te zahvaljujući njoj omogućen je razvoj pozitronske emisijske tomografije[27] i jednofotonske emisijske tomografije[28] što se smatra početkom moderne ere nuklearne medicine.[29]

Slika 9: Anger kamera i njen tvorac Hal Anger. Preuzeto iz [30]

Nuklearna medicina je medicinska specijalnost koja koristi radiofarmake za procjenu tjelesnih funkcije i za dijagnozu i liječenje bolesti. Radiofarmak prilikom raspada emitira gama zrake ili fotone visoke energije, a energija tih gama zraka ili fotona je takva da značajan broj izlazi iz tijela a da se pritom ne raspršii ili oslabi. Iz tog razloga, za generiranje slike o distribuciji

radiofarmaka, koriste se gama kamere koje mogu zabilježiti gama zrake ili fotone. Postoje dvije klase snimanja u nuklearnoj medicini a to su :

- Jednofotonsko snimanje, u kojem se koristi jednofotonska emisijska tomografija (SPECT)
- Snimanje pozitrona, u kojem se koristi pozitronska emisijska tomografija (PET)

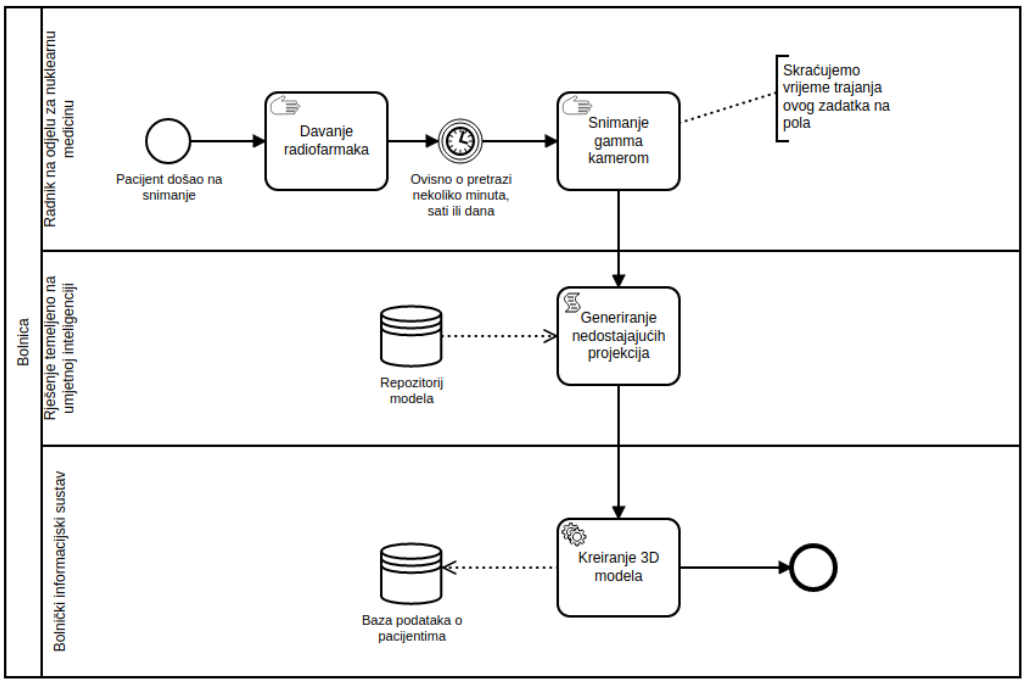
Nedostaci nuklearni medicine su ti što procesi snimanja obično dugo traju te je za to vrijeme potrebno da pacijent bude nepomičan kako bi se dobili najbolji mogući rezultati.

## 3.2 Jaszczak fantom

Jaszczakov fantom koristi se za evaluaciju performansi SPECT sistema te osim u tim slučajevima može se koristiti za evaluaciju programa za snimanje pacijenta i u istraživačke svrhe. Sastoji se od glavnog cilindra, spremnika, izrađenog od akrila s nekoliko umetaka. Glavni cilindar ima sljedeće specifikacije : unutarnji promjer 21.6 cm, unutarnja visina 18.6 cm i debljina stijenke 3.2 cm. Svaki fantom ima šest čvrstih sfera i šest kompleta hladnih šipki. Sfere se koriste za mjerenje kontrasta slike, dok se šipke koriste za ispitivanje razlučivosti slike u SPECT sustavima.[31]

Slika 10: Prikaz Jaszczak fantoma. Preuzeto s [32]

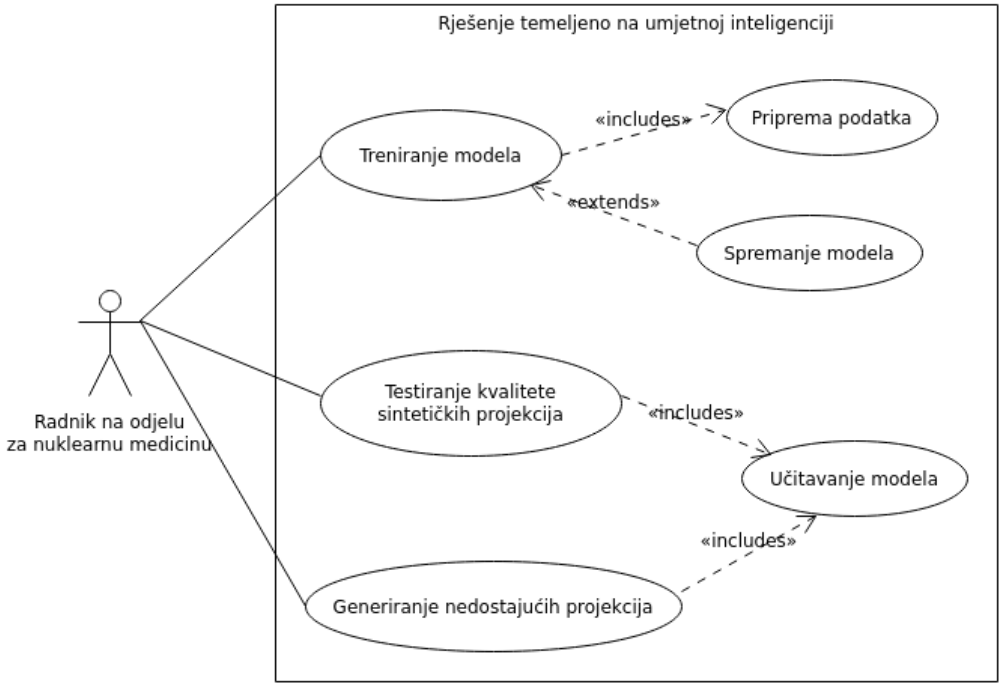
# 4 Generiranje sintetičikih projekcija




Slika 11: Prikaz procesa snimanja pomoću predloženog rješenja temeljenog na umjetnoj inteligenciji u jednofotonskoj emisijskoj tomografiji

Prije nego započnemo s detaljnim pregledom implementacije generiranja sintetičkih projekcije prikazat ćemo kako izgleda proces snimanja u jednofotonskoj emisijskoj tomografiji. Na slici 11 prikazan je dijagram poslovnog procesa snimanja te iz njega možemo vidjeti kako se naše rješenje uklapa u isti. Proces započinje kada pacijent dolazi na snimanje u odjel za nuklearnu medicinu te mu radnik daje radiofarmak. Nakon tog zadatka potrebno je čekati određeno vrijeme, ovisno o tipu pretrage može varirati od nekoliko minuta,sati ili dana, da se radiofarmak apsorbira u tijelo nakon čega se može započeti sa snimanjem gama kamerom. Želimo smanjiti vrijeme trajanja snimanja gama kamerom tako da namjestimo kameru da preskače svaki drugi kut postižući time dvostruko kraće vrijeme snimanja. Pošto takvim načinom snimanja nemamo dovoljno informacija, da pomoću rekonstrukcijskih algoritama kreiramo 3D model visoke kvalitete, potrebno je prijeći na sljedeći zadatak, a to je generiranje nedostajućih projekcija. U tom zadatku iz repozitorija modela uzimamo model, istreniran za specifičan organ, koji iz polovičnog seta projekcija, dobivenog tijekom snimanja, generira sve nedostajuće projekcije čime dobivamo potpuni set projekcija od kojih će pola

biti sintetski generirano a pola dobiveno direktno iz kamere. Takav se set prosljeđuje za sljedeći zadatak koji je kreiranje 3D modela, pomoću kojeg liječnik može postaviti dijagnozu, za što se koriste rekonstrukcijski algoritmi kojima bolnički informacijski sustav raspolaže. Nakon što je 3D model kreiran sprema se u bazu podataka o pacijentima, tako da mu liječnik tijekom pregleda pacijenta može pristupiti, te time završava proces snimanja.




Slika 12: Prikaz diagrama slučaja upotrebe predloženog rješenja temeljenog na umjetnoj inteligenciji

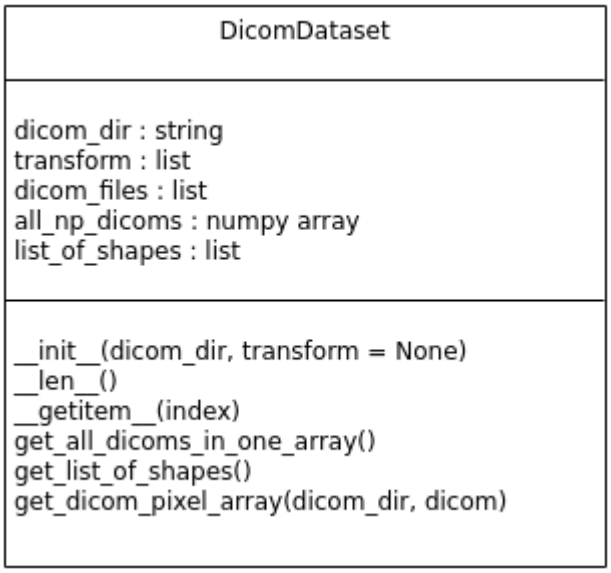
Nakon što smo prikazali kako se naše rješenje temeljeno na umjetnoj inteligenciji uklapa u poslovni proces snimanja u jednofotonskoj emisijskoj tomografiji prikazat ćemo kakav je slučaj upotrebe istog. Slika 12 predstavlja diagram slučaja upotrebe na kojem je moguće uočiti različite funkcije predloženog rješenja temeljenog na umjetnoj inteligenciji koje akter, u ovome dijagramu to je radnik na odjelu za nuklearnu medicinu, može izvršiti. Akter može trenirati modele što zahtjeva da se svaki put izvrši priprema podataka, koji bi trebali biti potpuni setovi projekcija određenog organa. Ako je akter zadovoljan rezultatima treninga moguće je proširiti taj slučaj upotrebe tako što se omogućuje spremanje modela. Također nudi se mogućnost za testiranje kvalitete sintetičkih projekcija različitim testovima kvalitete slika što zahtjeva učitavanje istreniranog modela na kojem će se ti testovi izvršiti.

Naposljetku možemo primijetiti i zadnji slučaj upotrebe, ujedno i najvažniji, a to je generiranje nedostajućih projekcija, već prikazan kao zadatak u poslovnom procesu, koji također zahtjeva učitavanje modela, pomoću kojega će se nedostajuće projekcije generirati.

U ovome poglavlju obradit ćemo kompletan proces koji je potreban za generiranje sintetičkih projekcija u jednofotonskoj emisijskoj tomografiji. Bit će prikazano kako su podaci pripremljeni za treniranja modela, koji su modeli izrađeni i zašto, kako je tekao proces treniranja modela, kakva je kvaliteta projekcija generiranih modelom i na kraju će biti predstavljeno kako se pomoću istreniranih modela generira potpuni set projekcija. Programski jezik korišten u tu svrhu je Python, a neke od glavnih biblioteka su Pytorch[33], Pytorch Image Quality[34] (PIQ), Pydicom[35] i Numpy[36].

## 4.1 Set podataka

Priprema podataka jedan je od najvažniji koraka prije samog treniranja nih mreža ili bilo kojeg drugog modela strojnog učenja. Prilikom izrade ovog rješenja temeljenog na umjetnoj inteligenciji na raspolaganju smo imali 10 potpunih setova projekcija Jaszczak fantoma od kojih šest setova sadrži 256 a četiri 360 projekcija crno bijele boje matrica veličine 128x128, no valja napomenuti da bi tijek pripreme bio isti i u slučaju potpunih setova projekcija nekog određenog organa. Tih 10 setova potpunih projekcija nalazile su se unutar DICOM (*eng. Digital Imaging and Communications in Medicine*) datoteka, što je ujedno i standard za upravljanje medicinskim slikovnim informacijama [37]. Kako bi mogli koristiti podatke iz DICOM-a za treniranje naših modela, potrebno je projekcije izvući iz datoteke te ih uskladiti sa standardom za Pytorch modele. Za tu svrhu kreirana je klasa DicomDataset koja nasljeđuje Dataset klasu Pytorch biblioteke.




Slika 13: Prikaz klase DicomDataset

Slika 13 prikazuje od kojih atributa i metoda se sastoji klasa DicomDataset. Pošto su svi atributi i metode javni njihova vidljivost nije posebno naglašena u prikazu. Prilikom instanciranja klasa prima dva argumenta, prvi *dicom_dir* i drugi opcionalni *transform*, koji se spremaju u istoimene atribute u klasi. Slijedi objašnjenje svakog atributa.

- *dicom_dir*. String koji sadrži lokaciju direktorija u kojem se nalaze DICOM datoteke

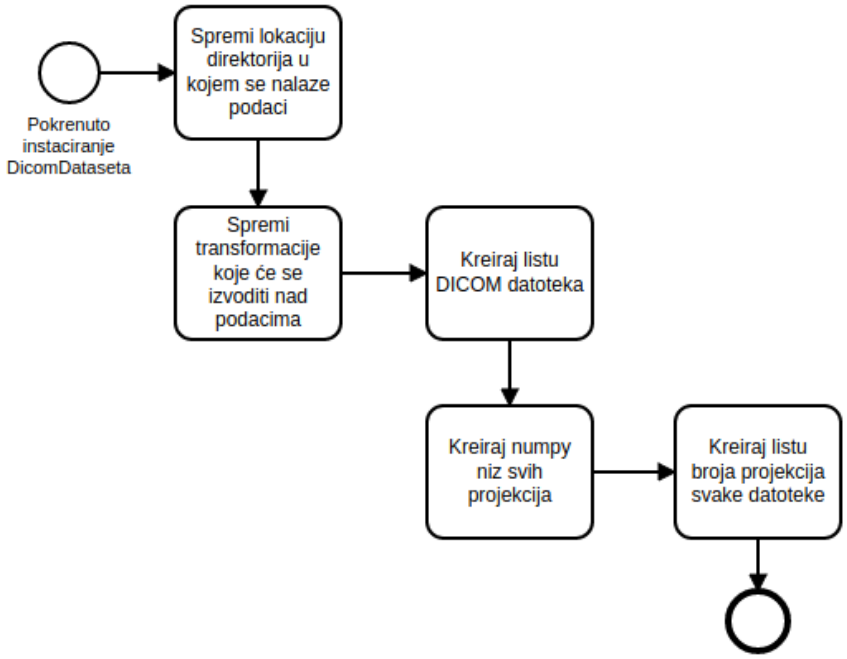
- *transform*. Lista transformacija koje će se obaviti nad podacima prilikom dohvaćanja za model
- *dicom_files*. Lista svih DICOM datoteka učitanih iz direktorija
- *all_np_dicoms*. Numpy niz svih projekcija iz svih DICOM datoteka na jednome mjestu
- *list_of_shapes*. Lista zbrojeva ukupnog broja projekcija iz svake DICOM datoteke po redoslijedu kako su učitavane, što nam pomaže da odredimo početak i kraj pojedinog seta projekcija prilikom dohvaćanja određene projekcije

Nakon što smo objasnili atribute klase DicomDataset slijedi objašnjenje metoda. Kao što možemo primijetiti iz slike 13 postoji šest metoda unutar klase od kojih su tri takozvane dunder metode, koje se mogu prepoznati po duplim podvlakama.

- *__init__*. Konstruktor metoda koja se poziva prilikom instanciranja klase, unutar nje se popunjuju svi atributi klase
- *__len__*. Vraća ukupan broj indeksa podataka unutar našeg seta podataka, a taj broj izračuna tako da uzme veličinu *all_np_dicoms* i umanji je za 1 jer numpy array počinje s indeksom 0. Obavezna je metoda u svakoj Pytorch Dataset klasi.
- *__getitem__*. Kao argument prima *index* koji služi za određivanje pozicije projekcija koje će se predati u model. Unutar nje prvo se određuju tri indeksa za projekcije, očekivani, očekivani - 1 i očekivani + 1. Pomoću *list_of_shapes* provjera se je li pozicija očekivanog indeksa na početku, kraju ili negdje u sredini seta projekcija te nakon toga pomoću ta tri indeksa iz *all_np_dicoms* uzimaju tri numpy niza koji predstavljaju očekivanu projekciju, prethodnu projekciju i sljedeću projekciju. Nakon što su numpy nizovi izabrani nad njima se rade transformacije, ukoliko su predane prilikom instanciranja, kako bi bili kompatibilni za treniranje Pytorch modela. Naposljetku takvi transformirani nizovi, sada tenzori, vraćaju se kao rezultat metode. Ova metoda također je obavezna u svakoj Pytorch Dataset klasi.
- *get_dicom_pixel_array*. Metoda prima dva argumenta *dicom_dir* koji se nalazi u argumentima klase i *dicom* koji označava DICOM datoteku. Unutar nje se pomoću Pydicom-a datoteka učita, nakon čega se izvuče numpy niz koji sadrži sve projekcije te datoteke. Metoda vraća numpy niz svih projekcija datoteke.

- *get_list_of_shapes*. Služi za popunjavanje *list_of_shapes* atributa. Stvara listu broja projekcija tako što prolazi kroz DICOM datoteke, pomoću *get_dicom_pixel_array* iz svake datoteke izvlači broj projekcija te ih zbraja s brojem projekcija prethodne. Kao rezultat vraća listu.

- *get_all_dicoms_in_one_array*. Služi za popunjavanje *all_np_dicoms* atributa. Unutar nje se prolazi kroz sve DICOM datoteke te pomoću *get_dicom_pixel_array* izvlače se numpy nizovi projekcija koji se prvo skaliraju u raspon od 0 do 255 s tipom niza uint8 te se spremaju u listu. Na kraju ta se lista pretvara u numpy niz koji se kao rezultat vraća iz funkcije.




Slika 14: Prikaz procesa koji se odvija tijekom instanciranja klase DicomDataset

Proces koji se odvija prilikom instanciranja klase DicomDataset prikazan je slikom 14. Možemo primijetiti iz slike da prvi koraci u tom procesu jesu spremanje lokacije direktorija u kojem se nalaze podaci te spremanje transformacije koje će se izvoditi nad podacima. Atributi koji će se pritom popunjavati jesu *dicom_dir* i *transform* te se njihove vrijednosti predaju kao argumenti prilikom instanciranja klase. Sljedeći zadatak je kreiranje liste DICOM datoteke što se radi pomoću standardne Python biblioteke os i njene

funkcije *listdir*. Kad imamo spremnu listu DICOM datoteke započinje zadatak kreiranja numpy niza svih projekcija za čije se izvršavanje koristi metoda *get_all_dicoms_in_one_array*. Nakon kreiranja numpy niza svih projekcija preostaje još kreirati listu broja projekcija svake DICOM datoteke što se radi pomoću metode *get_list_of_shapes* te izvršavanjem tog zadatke završava proces instanciranja klase DicomDataset.

## 4.2 Arhitekture modela za generiranje projekcija

Kako bi generirali sintetičke projekcije potrebno je izraditi modele koji će biti specijalizirani za taj zadatak. Različite arhitekture modela davat će različite rezultate, pa tako za dobivanje najboljih rezultata prilikom generiranja projekcija, arhitektura modela mora biti prilagođena zadatku za koji će se koristiti, što znači da sama arhitektura ne smije biti niti previše kompleksna niti pre jednostavna jer će u takvim slučajevima model generirati nedovoljno dobre projekcije. Iz tog razloga, za naše rješenje temeljeno na umjetnoj inteligenci, morali smo kreirati model prilagođen generiranju sintetičkih projekcija.

U ovome potpoglavlju obradit će se modeli koji su se koristili za testiranje kvalitete projekcija i kasnije u generiranju sintetičkih projekcija. Postoje dva modela koja će se obraditi, prvi je Baseline a drugi DCNN.

### 4.2.1 Baseline

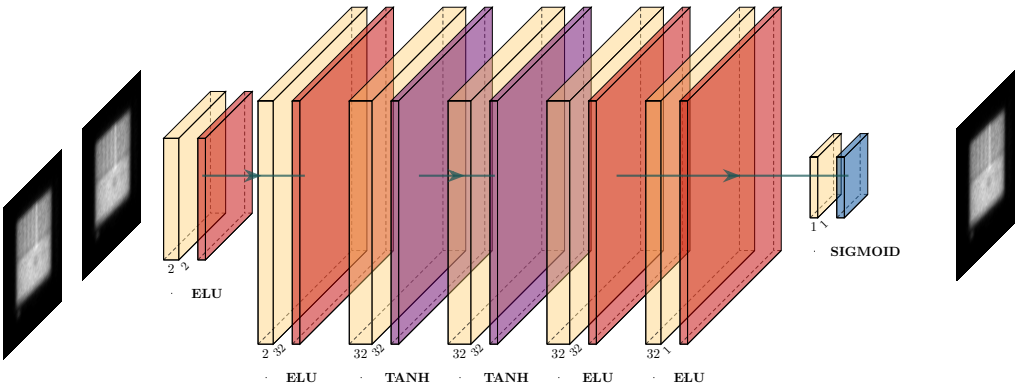
Baseline model kreiran je kao polazišna točka te služi za usporedbu s DCNN modelom. Iz tog razloga Baseline model je jednostavan i radi po sljedećoj formuli.

$$projekcija_{očekivana} = \frac{projekcija_{prethodna} + projekcija_{sljedeća}}{2}$$

Možemo primijetiti da se radi o aritmetičkoj sredini projekcija pozicijski prije i poslije očekivane projekcije. Takav model ne zahtijeva treniranje te se njegov rad može opisati u sljedećih nekoliko koraka :

1. Za argumente primi dva tenzora
2. Pretvori tenzore u numpy nizeve
3. Izračunaj aritmetičku sredinu dva numpy niza te spremi rezultat
4. Pretvori rezultat u tenzor
5. Dodaj tenzoru još jednu dimenziju kako bi bio kompatibilan za testiranje kvalitete
6. Vrati tenzor kao rezultat

### 4.2.2 DCNN




Slika 15: Prikaz DCNN arhitekture

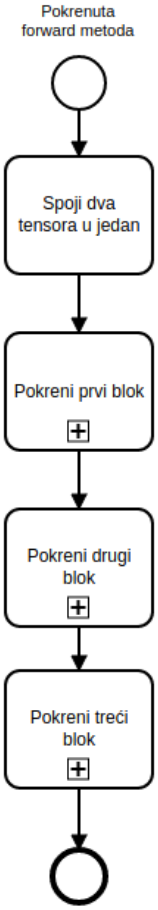
DCNN (*eng. Deep Convolutional Neural Network*) je predloženi Pytorch model posebno prilagođen generiranju sintetičkih projekcija. Na slici 15 prikazana je arhitektura modela. Iz nje možemo vidjeti da model na ulazu prima dvije originalne projekcije, a na izlazu daje jednu sintetički generiranu. Model se sastoji od sedam konvolucijskih slojeva to jest Pytorch *Conv2d* objekata, na slici 15 označenih žutom bojom, nakon kojih dolaze aktivacijske funkcije. Tih sedam konvolucijskih slojeva podijeljeno je u tri bloka od kojih prva dva bloka sadrže dva konvolucijska sloja i dvije aktivacijske funkcije dok treći block sadrži po tri sloja i aktivacijske funkcije. Slijedi opis svakog bloka :

- Prvi blok sadrži ELU aktivacijsku funkciju nakon svakoj konvolucijskog sloja, na slici 15 označenih crvenom bojom, a njegov prvi konvolucijski sloj prima tensor s dva kanala te na izlazu daje tensor s dva kanala dok drugi sloj prima tensor s dva te na izlazu daje tenzor s 32 kanala.

- Drugi block sadrži Tanh aktivacijsku funkciju, označenu ljubičastom bojom, a njegovi konvolucijski slojevi na ulazu primaju te na izlazu daju tenzor s 32 kanala.

- Treći blok sadrži tri konvolucijska sloja te njegov prvi sloj prima i daje tenzor s 32 kanala, drugi sloj prima tenzor s 32 te na izlazu daje tenzor s jednim kanalom i treći sloj koji prima i daje tenzor s jednim kanalom. Nakon prva dva konvolucijska sloja nalazi se ELU aktivacijska funkcija dok nakon zadnjeg sloja slijedi Sigmoid aktivacijska funkcija, na slici 15 označena plavom bojom.

Važno je napomenuti da svaki konvolucijski sloj još prima argumente, *kernel size* od (3,3), *stride* od (1,1) i *padding* od 1, kojima su vrijednosti specifično određene kako bi oblik podataka koji prolazi kroz njih ostao nepromijenjen. Odabir različitih aktivacijskih funkcija u jednom modelu na prvu može izgledati ne konvencionalno, no tijekom izgradnje i testiranje različitih arhitektura modela upravo je ova raspodjela aktivacijski funkcija i ovaj broj konvolucijskih slojeva davao najkonzistentnije rezultate tijekom treniranja i testiranja.



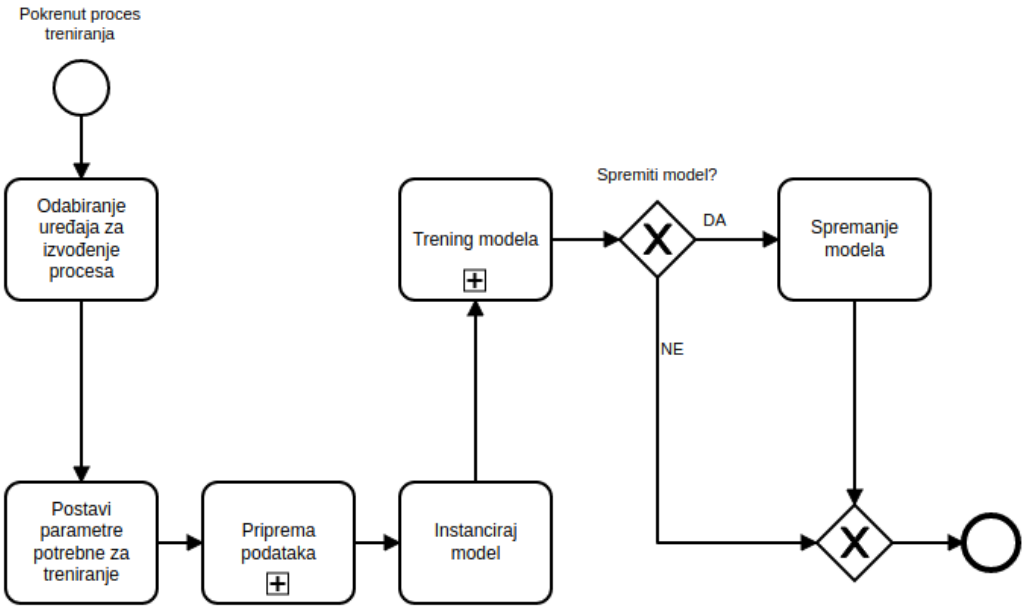

Slika 16: Prikaz rada *forward* metode

Treba spomenuti da se unutar ove arhitekture nalazi *forward* metoda unutar koje definiramo cijelu strukturu te kako će se izvršavati naš model prilikom svakog njegov poziva. Rad te metode može se primijetiti iz slike 16. Kao argumente prima dva jedno kanalna tenzora koja spaja u jedan tenzor s dva kanala te je to prvi zadatkom unutar procesa. Takav tensor se prvo

prosljeđuje prvom bloku, njegov izlaz drugom bloku te finalno se izlaz drugog bloka prosljeđuje trećem bloku. potprocesi koji se izvršavaju unutar svakog pojedinog bloka specificirani su prilikom opisa svakog bloka. Proces završava izlaskom tensora iz trećeg bloka time dobivajući rezultat *forward* metode što je sintetički generirana slika.

## 4.3 Treniranje modela

Treniranje modela esencijalano je za korištenje modela neuronskih mreža.




Slika 17: Prikaz kompletnog procesa treninga

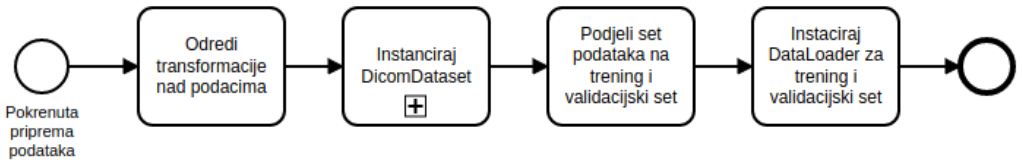
Kompletan proces treninga možemo vidjeti u slici 17. Sam proces počinje kad se pokrene skripta te se na početku, ovisno o dostupnosti grafičke kartice, odabire hoće li se daljnji dijelovi procesa, koji su za to kompatibilni, izvodi na grafičkoj kartici ili procesoru. Nakon odabira uređaja na kojem će se izvodi određeni dijelovi procesa slijedi postavljanje parametara potrebnih za treniranje. U tom zadatku treba postaviti sljedeće parametre :

- **Veličina serija**(*eng. batch size*). Hiperparametar služi za određivanje koliko će se serija podataka predati modelu u jednoj iteraciji prije ažuriranje parametara modela. U našem slučaju postavljen je na 20.
- **Stopa učenja**(*eng. learning rate*). Hiperparametar koji kontrolira koliko će se model mijenjati svaki put kada se ažuriraju parametri modela. Postavljen je na 0.0002
- **Broj epoha**. Određuje koliko će se puta proći kroz cijeli set podataka. Postavljeno na 100.
- **Funkcija gubitka**(*eng. loss function*). Računa grešku između izlaza modela i ciljane vrijednosti. Cilj je minimizirati. U našem treningu

korišten je SSIMLoss, iz PIQ biblioteke, s parametrima *data_range* 1.0, *kernel_size* 11, *k1* 0.01 i *k2* 0.03 .

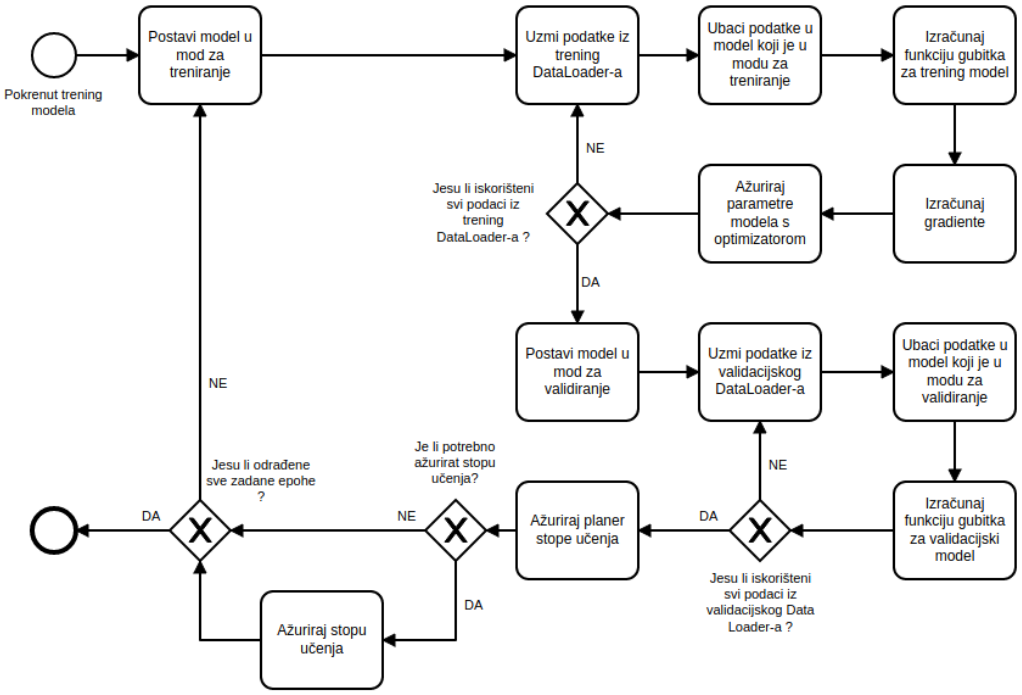
- **Optimizator**. Pomoću njega ažuriramo parametre modela. Korišten je Adam.
- **Planer stope učenja**. Pomoću njega se ukoliko, nakon određenog broja epoha, ne dolazi do smanjenja prosječne vrijednosti validacijske funkcije gubitka smanjuje stopu učenja. Korišten je ReduceLROnPlateau s parametrima *factor* 0.643215, *patience* 8, *threshold_mode* "abs" i *threshold* 0.0001

Nakon što smo odredili sve potrebne parametre za treniranje slijedi priprema podataka za trening. Pod proces pripreme podataka prikazan je u slici 18. On započinje zadatkom određivanja transformacija koje će se izvršiti nad podacima. Transformacije koje su se koristile u treningu našeg DCNN modela, implementirane pomoću Torchvision biblioteke, su *ToTensor*, za pretvaranje podataka u tenzore, i *Normalize* sa parametrima (0.5,) (0.5,), za normalizaciju podataka u raspon od -1 do 1 kako bi pospješili treniranje modela. Nakon određivanja transformacije predaju se zajedno s lokacijom DICOM datoteka za potrebe instanciranja DicomDataseta, a taj potproces prikazan je slikom 14. Dobiveni set podataka potom je potrebno podijeliti na trening i validacijski set, što je napravljeno pomoću funkcije *random_split* u omjeru 85% za trening i 15% validacijski set. Pomoću takvih podijeljenih setova podataka potrebno je instancirati DataLoader za svaki. DataLoader služi za iteraciju po setu podataka i izbacivanje podataka u serijama, određenih hiperparametrom **veličina serija**. Valja napomenuti da je parametar za DataLoader *shuffle* postavljen na True, iz razloga da se podaci promiješaju nakon svake epohe što doprinosi boljem modelu.




Slika 18: Prikaz pod procesa pripreme podataka

Nakon što je završen pod proces pripreme podataka slijedi instanciranje modela, u našem slučaju DCNN modela, koji će se trenirati u sljedećem pod procesu. Pod proces treninga modela prikazan je slikom 19. Može se primijeti da unutar pod procesa treninga modela imamo petlje, jednu glavnu,




Slika 19: Prikaz pod procesa trening modela

koja prolazi po epohama, i dvije, redoslijedom bitne, manje petlje od kojih je prva za treniranje a druga za validiranje. Pod proces započinje ulaskom u glavnu petlju te se prvo model stavlja u mod za treniranje nakon čega slijedi petlja sa sljedećim redoslijedom izvršavanja:

1. Uzmi podatke iz trening DataLoader-a, što je serija podataka za prethodnu, sljedeću i očekivanu projekciju

2. Podaci se ubacuju u model koji je u modu za treniranje, a to su serija podataka za prethodnu i sljedeću projekciju

3. Izračunaj funkciju gubitka za trening model, što se radi tako da se u funkciju gubitka preda izlaz modela i serija podataka za očekivanu projekciju

4. Izračunaj gradiente, što se radi pomoću *backwards* metode funkcije gubitka

5. Ažuriraj parametre modela s optimizatorom, što se postiže koristeći *step* metodu optimizatora

Petlja se ponavlja sve dok se ne prođu svi podaci iz trening DataLoader-a. Nakon što se prođu svi podaci iz trening DataLoader-a, model se postavlja u mod za validaciju te započinje sljedeća petlja sa redoslijedom izvršavanja :

1. Uzmi podatke iz validacijskog DataLoader-a
2. Podaci se ubacuju u model koji je u modu za validiranje
3. Izračunaj funkciju gubitka za validacijski model

Takva petlja ponovo se ponavlja sve dok se ne prođu svi podaci iz validacijskog DataLoader-a. Ako su svi podaci iz validacijskog DataLoader-a obrađeni prelazi se na sljedeći zadatak koji je ažuriranje planera stope učenja. Ako je potrebno ažurirati stopu učenja to se izvrši. Nakon toga ponavljamu glavnu petlju sve dok je ne izvršimo onoliko puta koliko je zadano u **broju epoha**. Po završetku zadnje epohe, to jest zadnjeg prolaska kroz glavnu petlju, pod proces treninga modela završava.

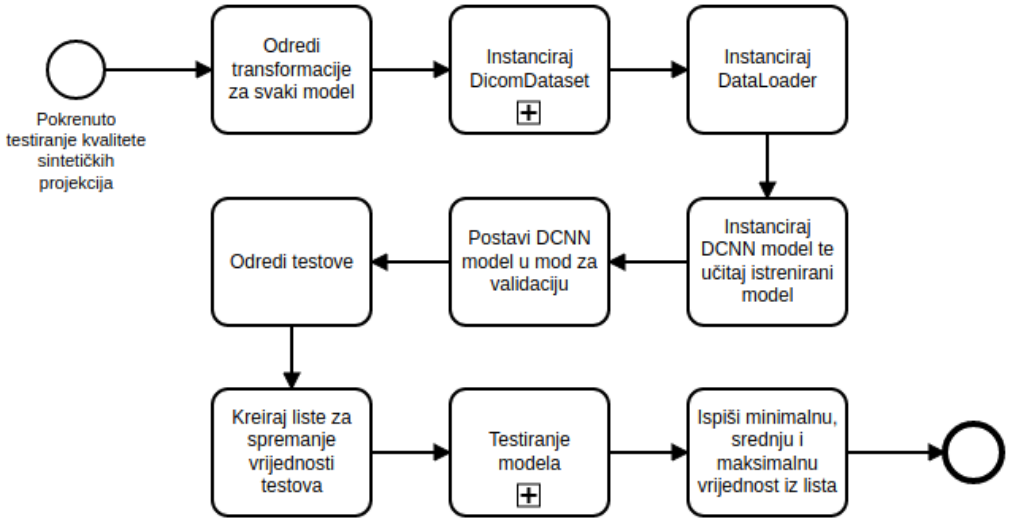
Nakon što je pod proces treninga modela završen moguće je spremiti istreniran model te time završava proces treniranja.

## 4.4 Testiranje kvalitete sintetičkih projekcija

Procjena kvalitete sintetičkih projekcija izvodi se nad Baseline i DCNN modelom nakon njegovog treniranja. Odabrani su sljedeći testovi za kvalitetu slike :

- SSIM(*eng. Structural Similarity Index Measure*)[38].
- VIF(*eng. Visual Information Fidelity*)[39].

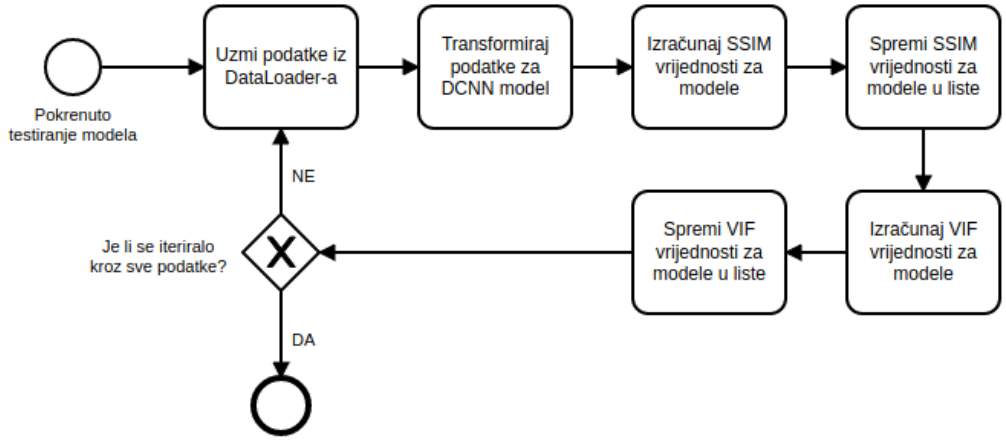
Testiranje se provodilo na jednom odvojenom setu potpunih projekcija Jaszczakova fantoma, koji nije bio korišten prilikom trening faze modela.




Slika 20: Prikaz procesa testiranja kvalitete sintetičkih projekcija

Proces testiranja kvalitete sintetičkih projekcija započinje tako što se odrede transformacije za podatke koje su potrebne svakom modelu te je prikazan slikom 20. U tom zadatku kreiraju se dvije transformacije, prva koja služi Baseline modelu te će pretvarati podatke u tenzore i druga koja služi DCNN modelu te sadržava transformacije koje su se provodile prilikom treniranja modela. Nakon toga slijedi instanciranje DicomDataset klasa, kojoj će se kao argumenti predati lokacija testnog seta potpunih projekcija i prva transformacija za Baseline model, te se ona prosljeđuje za instanciranje DataLoader-a koji ima parametar veličine serija 1. Nakon što je to odrađeno, instancira se DCNN model te se učita prethodno istrenirani model. Potrebno je postaviti DCNN model u mod za validaciju u sljedećem zadatku kako bi dobivali dobre rezultate. Sljedeći korak je određivanje testova koji će se

koristiti, a kao što je spomenuto ranije u tekstu koristit će se SSIM i VIF funkcije gubitka iz PIQ biblioteke. Pošto koristimo funkcije gubitka potrebno izračunati 1 - vrijednost funkcije gubitka kako bi dobili stvarnu vrijednost. Prije samog testiranja potrebno je kreirati liste za spremanje vrijednosti testova. U tom zadatku kreirat će se odvojena lista za svaki model i svaki test.




Slika 21: Prikaz potprocesa testiranja modela

Nakon toga slijedi potproces testiranja modela, prikazan slikom 21, unutar kojeg se nalazi petlja koja se izvršava sljedećim redoslijedom :

1. Uzmi podatke iz DataLoader-a
2. Transformiraj podatke za DCNN model
3. Izračunaj SSIM vrijednosti za modele
4. Spremi SSIM vrijednosti za modele u liste
5. Izračunaj VIF vrijednosti za modele
6. Spremi VIF vrijednosti za modele u liste

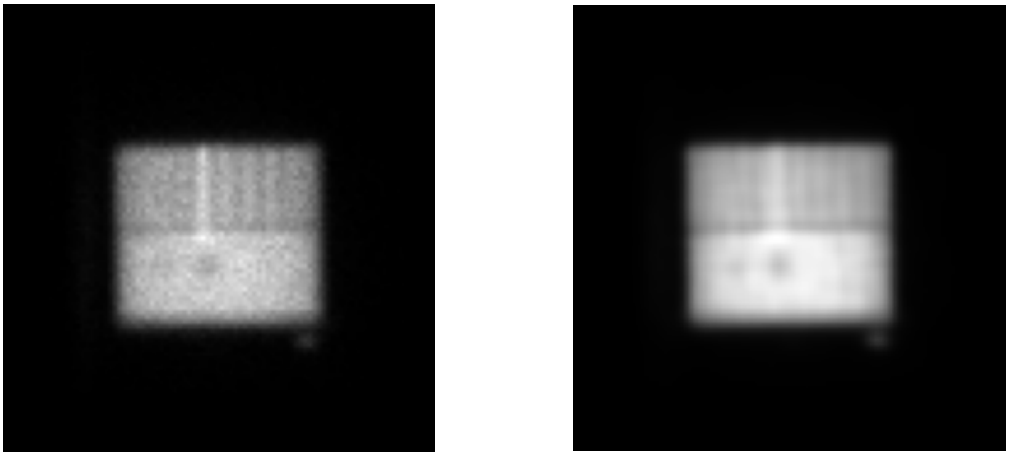
Prilikom računanja SSIM i VIF vrijednosti prvo se računaju vrijednosti za Baseline a potom za DCNN model te se izračunate vrijednosti spremaju u odvojene liste za svaki model i test. Kad se iterira kroz sve podatke iz DataLoader-a petlja time završava te je potproces završen. Nakon završetka potprocesa testiranja modela ispisuju se minimalna, srednja i maksimalna

vrijednost iz lista čijim ispisom završava proces testiranja kvalitete sintetičkih projekcija.

Rezultati provedenog testiranja prikazani su tablicom 1 te iz nje možemo vidjeti da DCNN model u slučaju oba testa generira malo kvalitetnije sintetičke projekcije.

| | **SSIM** | | | **VIF** | | |
|---|---|---|---|---|---|---|
| | Prosječno | Minimalno | Maksimalno | Prosječno | Minimalno | Maksimalno |
| **Baseline** | 0.9656 | 0.9541 | 0.9703 | 0.6398 | 0.5738 | 0.6565 |
| **DCNN** | 0.9695 | 0.9585 | 0.973 | 0.6647 | 0.5948 | 0.6851 |

Tablica 1: Rezultati testova kvalitete sintetčkih projekcija za Baseline i DCNN model



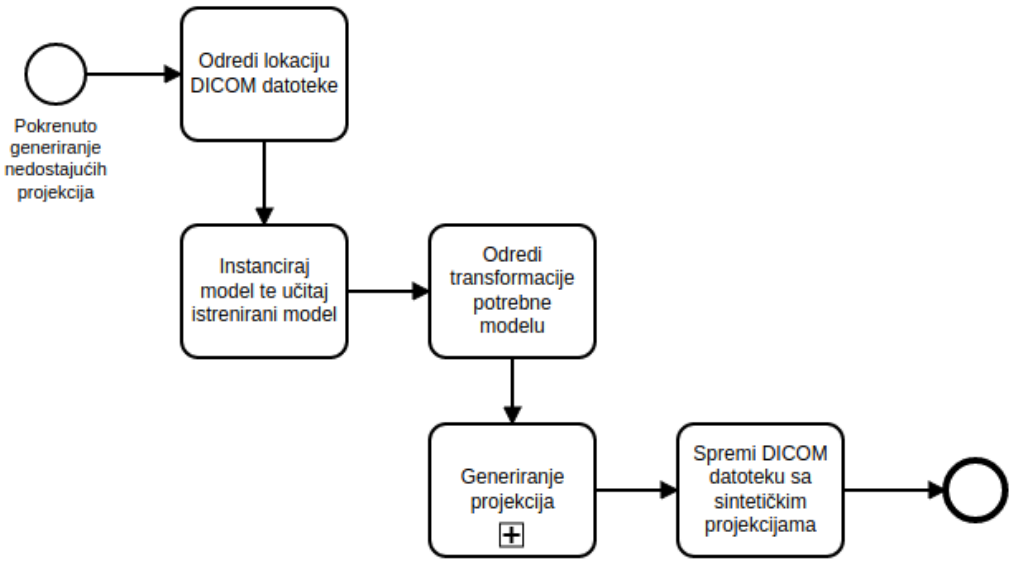
(a) Originalna (b) Sintetička

Slika 22: Prikaz originalne i pomoću DCNN modela sintetički generirane projekcije

Na slici 22 možemo uočiti razliku između originalne 22a i sintetički generirane projekcije 22b, generirane koristeći DCNN model. Iako je uočljiva razlika između originalne i sintetičke projekcije, sve bitne karakteristike iz 22a, kao što su sfere i šipke iz Jaszczakova fantoma, također se mogu uočiti u 22b.

## 4.5 Generiranje potpunog seta projekcija

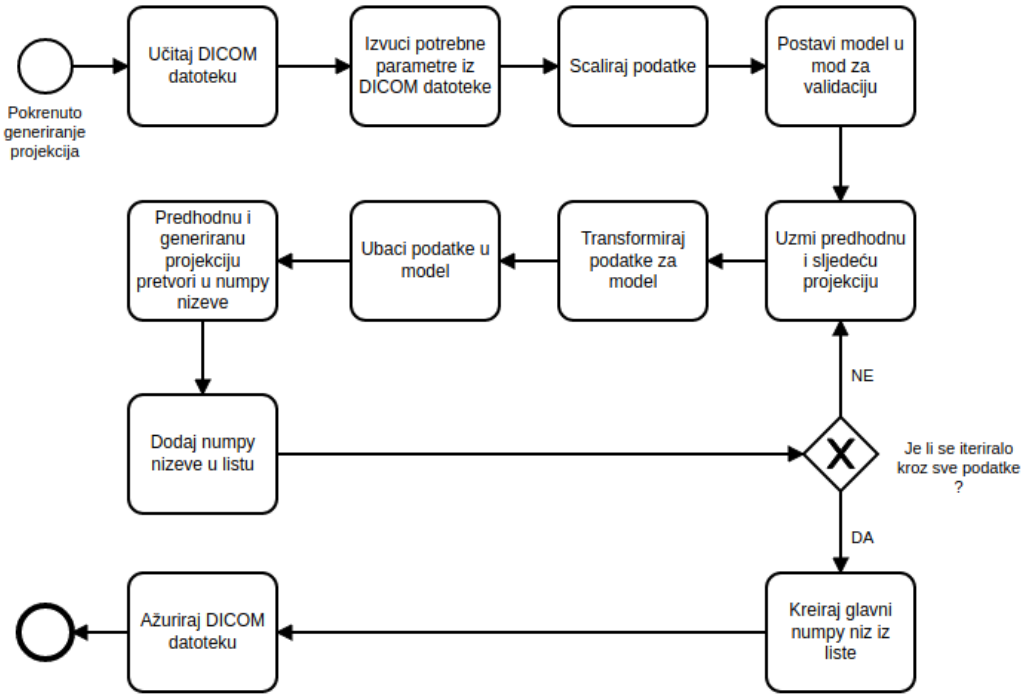
Koristeći model, čijim smo rezultatima testova kvalitete projekcija zadovoljni, možemo generirati nedostajuće projekcije. Proces generiranja nedostajućih projekcija prikazan je slikom 23. Iz nje možemo vidjeti da proces započinje tako što odredimo lokaciju DICOM datoteke na uređaju. Slijedi instanciranje modela te učitavanje modela koji je davao dobre rezultate tijekom provođenja testova kvalitete. Nakon toga se određuju transformacije potrebne modelu, koje trebaju biti iste kao i one koje su se koristile tijekom treniranja. Kad su prethodni koraci odrađeni poziva se funkcija za generiranje projekcija kojoj se kao argumenti predaju lokacija DICOM datoteke, model koji će se koristiti i transformacije koje su potrebne modelu.




Slika 23: Prikaz kompletnog procesa generiranja nedostajućih projekcija

Funkcija generiranja projekcija prikazana je u pod procesu generiranja projekcija na slici 24. Potrebno je prvo učitati DICOM datoteke iz njene lokacije pomoću Pydicom biblioteke. Nakon toga slijedi izvlačenje potrebnih parametara iz iste. Ti parametri su sljedeći :

- Numpy niz koji sadrži sve projekcije unutar DICOM datoteke
- Maksimalna i minimalna vrijednost piksela u numpy nizu
- Tip numpy niza




Slika 24: Prikaz potprocesa generiranja projekcija

Unutar sljedećeg zadatka potrebno je skalirati numpy niz da bude kompatibilan s potrebama Pytorch model, što znači da će se vrijednost svih piksela skalirati u raspon od 0 do 255 te će mu se promijeniti tip u uint8, kao što se to radilo unutar DicomDataset klase. Također je potrebno postaviti model u mod za validaciju, kao što je to bio slučaj u testiranju kvalitete projekcija, jer inače nećemo dobivati dobre rezultate. Nakon toga slijedi petlja generiranja koja se može prikazi sljedećim redoslijedom :

1. Uzmi prethodnu i sljedeću projekciju, te im dodaj jedna dimezija za kanal boje
2. Transformiraj projekcije, naknadno im dodaj još jednu dimenzija kako bi konačni tenzor bio 4D jer model očekuje tenzor koji ima veličinu serije, broj kanala, širinu i visinu
3. Ubaci podatke u model
4. Uzimi prethodni i generirani tenzor te ga pretvori natrag u numpy niz, pritom mijenjajući tip i skalirajući vrijednosti piksela u rasponu minimalnih i maksimalnih originalnih vrijednosti piksela prethodno zabilježenih

5. Dodaj numpy nizeve u listu, pritom pazeći na redoslijed, prvo se dodaje prethodna a nakon nje generirana

Ova petlja ponavlja se sve dok se ne prođu svi podaci, odnosno sve projekcije unutar DICOM datoteke. Nakon završetka petlje, te popunjavanja liste s numpy nizovima, iz liste se kreira glavni numpy niz. Jedino što je preostalo prije završetka pod procesa generiranja projekcija je ažuriranje DICOM datoteke. U tom zadatku zamjenjuje se numpy niz originala s glavnim numpy nizom, koji sadrži sintetičke projekcije, te promjena parametara *Rows*, *Columns* i *NumberOfFrames* s parametrima iz novog glavnog numpy niza, što se radi iz razloga kako bi svi parametri DICOM datoteke odgovarali novom numpy nizu.

Po završetku pod procesa generiranja projekcija preostalo je odraditi zadatak spremanja DICOM datoteke sa sintetičkim projekcijama čime i završava proces generiranja nedostajućih projekcija.

# 5 Zaključak

U ovo radu dali smo pregled korištenih tehnologija te smo prikazali implementaciju našeg rješenja temeljenog na umjetnoj inteligenciji za generiranje sintetičkih projekcija u jednofotonskoj emisijskoj tomografiji te rezultate kvaliteta tih projekcija za dva različita model. Koristeći ovakvo rješenje skratili bi vrijeme trajanja procesa snimanja gama kamerom na pola što bi doprinijelo ubrzanju cjelokupnog procesa snimanja te bi utjecalo na smanjenje lista čekanja za ista. Valja napomenuti da rješenje temeljeno na umjetnoj inteligenciji nije testirano na podacima određenih organa, što zahtjeva posebna pravna dopuštenja, no pretpostavljamo da bi i ti rezultati bili zadovoljavajući. Za daljnja istraživanja trenirali bi, a potom testirali, rješenje temeljeno na umjetnoj inteligenciji na podacima specifičnih organa, nadalje pokušali bi vidjeti kakvi će rezultati biti ako se poveća razmak između kuteve pod kojim je izvršeno snimanje jer pretpostavljamo da je Baseline model davao iznenađujuće dobre rezultate baš zbog malog razmaka između kuteva na testnom setu projekcija i veoma male kompleksnosti u projekcijama Jaszczakova fantoma, a mislimo da veći razmak u kutevima i povećanje kompleksnosti u projekcijama specifičnih organa ne bi previše utjecalo na DCNN model. Treba naglasiti da korištenjem tehnologija dubokog učenja moguće je utjecati i na ostale faktore koji su presudni u jednofotonskoj emisijskoj tomografiji, neki od njih bili bi korekcija gibanja tijekom snimanje koje utječe na kvalitetu projekcija te smanjivanje doze radiofarmaka koji se daje prije samog snimanja što bi utjecalo na kvalitetu projekcija no to je moguće korigirati sa spomenutom tehnologijom.

# Literatura


[1] Junchi Liu, Yongyi Yang, Miles N. Wernick, P. Hendrik Pretorius, and Michael A. King. Deep learning with noise-to-noise training for denoising in SPECT myocardial perfusion imaging. *Medical Physics*, 48(1):156–168, November 2020.

[2] Maximilian P Reymann, Tobias Würfl, Philipp Ritt, Bernhard Stimpel, Michal Cachovan, A Hans Vija, and Andreas Maier. U-net for spect image denoising. In *2019 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pages 1–2. IEEE.

[3] Qi Zhang, Jingzhang Sun, and Greta S. P. Mok. Low dose spect image denoising using a generative adversarial network, 2019.

[4] Albert Juan Ramon, Yongyi Yang, P. Hendrik Pretorius, Karen L. Johnson, Michael A. King, and Miles N. Wernick. Initial investigation of low-dose SPECT-MPI via deep learning. In *2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC)*. IEEE, November 2018.

[5] Albert Juan Ramon, Yongyi Yang, P. Hendrik Pretorius, Piotr J. Slomka, Karen L. Johnson, Michael A. King, and Miles N. Wernick. Investigation of dose reduction in cardiac perfusion SPECT via optimization and choice of the image reconstruction strategy. *Journal of Nuclear Cardiology*, 25(6):2117–2128, May 2017.

[6] Albert Juan Ramon, Yongyi Yang, P. Hendrik Pretorius, Karen L. Johnson, Michael A. King, and Miles N. Wernick. Improving diagnostic accuracy in low-dose SPECT myocardial perfusion imaging with convolutional denoising networks. *IEEE Transactions on Medical Imaging*, 39(9):2893–2903, September 2020.

[7] Julian Betancur, Lien-Hsin Hu, Frederic Commandeur, Tali Sharir, Andrew J. Einstein, Mathews B. Fish, Terrence D. Ruddy, Philipp A. Kaufmann, Albert J. Sinusas, Edward J. Miller, Timothy M. Bateman, Sharmila Dorbala, Marcelo Di Carli, Guido Germano, Yuka Otaki, Joanna X. Liang, Balaji K. Tamarappoo, Damini Dey, Daniel S. Berman, and Piotr J. Slomka. Deep learning analysis of upright-supine high-efficiency SPECT myocardial perfusion imaging for prediction of obstructive coronary artery disease: A multicenter study. *Journal of Nuclear Medicine*, 60(5):664–670, September 2018.

[8] Felisa J. Vázquez-Abad, Silvano Bernabel, Daniel Dufresne, Rishi Sood, Thomas Ward, and Daniel Amen. Deep learning for mental illness detection using brain SPECT imaging. In *Medical Imaging and Computer-Aided Diagnosis*, pages 17–26. Springer Singapore, 2020.

[9] Nguyen Thanh Trung, Nguyen Thai Ha, Nguyen Duc Thuan, and Dang Hoang Minh. A deeplearning method for diagnosing coronary artery disease using spect images of heart. *Journal of Science & Technology*, 144:022–027, 2020.

[10] Charalambos Chrysostomou, Loizos Koutsantonis, Christos Lemesios, and Costas N. Papanicolas. SPECT imaging reconstruction method based on deep convolutional neural network. In *2019 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. IEEE, October 2019.

[11] Wenyi Shao and Yong Du. Spect image reconstruction by deep learning using a two-step training method. *Journal of Nuclear Medicine*, 60(supplement 1):1353–1353, 2019.

[12] Kenta Sakaguchi, Hayato Kaida, Shuhei Yoshida, and Kazunari Ishii. Attenuation correction using deep learning for brain perfusion SPECT images. *Annals of Nuclear Medicine*, 35(5):589–599, March 2021.

[13] Trung Thanh Nguyen, Thanh Nguyen Chi, Minh Dang Hoang, Ha Nguyen Thai, and Thuan Nguyen Duc. 3d unet generative adversarial network for attenuation correction of SPECT images. In *2020 4th International Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom)*. IEEE, August 2020.

[14] Mahsa Torkaman, Jaewon Yang, Luyao Shi, Rui Wang, Edward J. Miller, Albert J. Sinusas, Chi Liu, Grant T. Gullberg, and Youngho Seo. Direct image-based attenuation correction using conditional generative adversarial network for SPECT myocardial perfusion imaging. In Barjor S. Gimi and Andrzej Krol, editors, *Medical Imaging 2021: Biomedical Applications in Molecular, Structural, and Functional Imaging*. SPIE, February 2021.

[15] Tobias Rydén, Martijn Van Essen, Ida Marin, Johanna Svensson, and Peter Bernhardt. Deep-learning generation of synthetic intermediate projections improves 177lu SPECT images reconstructed with sparsely acquired projections. *Journal of Nuclear Medicine*, 62(4):528–535, August 2020.

[16] Warren S. Mcculloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[17] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[18] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, 1969.

[19] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[20] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*. OReilly, 2020.

[21] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 07 2006.

[22] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[25] Eli Stevens, Luca Antiga, and Thomas Viehmann. *Deep Learning with PyTorch*. Manning, 2020.

[26] Healthwise Staff. Single photon emission computed tomography. https://myhealth.alberta.ca/Health/Pages/conditions.aspx?hwid=acc4790. Pristupljeno : 01.05.2021.

[27] Michael E. Phelps, Edward J. Hoffman, Nizar A. Mullani, and Michel M. Ter-Pogossian. Application of annihilation coincidence detection to transaxial reconstruction tomography. *Journal of Nuclear Medicine*, 16(3):210–224, 1975.

[28] David E. Kuhl, Roy Q. Edwards, Anthony R. Ricci, Robert J. Yacob, Thomas J. Mich, and Abass Alavi. The mark IV system for radionuclide computed tomography of the brain. *Radiology*, 121(2):405–413, November 1976.

[29] Simon R. Cherry, James A. Sorenson, and Michael E. Phelps. *Physics in nuclear medicine*. Saunders, 2012.

[30] William G. Myers. The anger scintillation camera becomes of age. *Journal of Nuclear Medicine*, 20(6):565–567, 1979.

[31] *Nuclear medicine physics: A handbook for teachers and students*. IAEA International Atomic Energy Agency, 2015.

[32] https://capintec.com/product/jaszczak-standard-spect-phantom/. Pristupljeno : 01.05.2021.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[34] Sergey Kastryulin, Dzhamil Zakirov, and Denis Prokopenko. PyTorch Image Quality: Metrics and measure for image quality assessment, 2019. Open-source software available at https://github.com/photosynthesis-team/piq.

[35] Darcy Mason, scaramallion, rhaxton, mrbean bremen, Jonathan Suever, Vanessasaurus, Guillaume Lemaitre, Dimitri Papadopoulos Orfanos, Aditya Panchal, Alex Rothberg, Joan Massich, James Kerns, Korijn van Golen, Thomas Robitaille, moloney, Matthew Shun-Shin, pawelzajdel, Blair Conrad, Markus Mattes, Simon Biggs, Félix C. Morency, Markus D. Herrmann, Hans Meine, Joseph Wortmann, Kevin S. Hahn,

Masahiro Wada, Ram Rachum, colonelfazackerley, ferdymercury, and huicpc0207. pydicom/pydicom: pydicom 2.1.2, December 2020.

[36] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[37] NEMA PS3 / ISO 12052. *Digital Imaging and Communications in Medicine (DICOM) Standard*. National Electrical Manufacturers Association, Rosslyn, VA, USA. (available free at http://medical.nema.org/).

[38] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[39] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, 2006.

# Popis slika

# Popis tablica