UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

**IZVEŠTAJ O PROJEKTU IZ DUBOKOG UČENJA**

Tehnički izveštaj

Studijski program: Računarstvo i informatika

Modul: Inženjerstvo podataka

Student: Mentor:

Aleksandar Stojković, br. ind. 1915 Prof. dr Aleksandar Milosavljević

Niš, septembar 2025. godina

Sadržaj

[**1.** **UVOD** 3](#_Toc208053232)

[**2.** **TEORIJSKI PREGLED** 4](#_Toc208053233)

[**2.1.** Uslovni GAN - pix2pix 7](#_Toc208053234)

[**3.** **SKUP PODATAKA** 10](#_Toc208053235)

[**4.** **IMPLEMENTACIJA** 11](#_Toc208053236)

[**5.** **REZULTATI** 18](#_Toc208053237)

[**6.** **LITERATURA** 21](#_Toc208053238)

## **UVOD**

U okviru projekta je demonstrirana implementacija uslovne generativne suparničke mreže na primeru translacije skica ljudskog lica na slike ljudskog lica. Projekat je realizovan sa ciljem demonstracije jedne od naprednijih tehnologija dubokog učenja i obučen je model kao proof-of concept, odnosno demonstracija ove tehnologije.

Generativne suparničke mreže predstavljaju tip mreža za generisanje podataka, to jest, tipično slika. Sastoje se od dva modela, prvi model koje generiše podatke i samim tim se naziva generator i drugi model koji treba da vrši procenu da li su u pitanju pravi ili generisani podaci. Funkcionišu tako što se putem min max naizmeničnog obučavanja i generator i diskriminator treniraju u međusobno suparničkom odnosu. Cilj ovakvog obučavanja je postizanje međusobnog unapređenja ovih modela sve dok diskriminator više nije u stanju da razluči prave od generisanih slika.

Za obučavanje je korišćen skup podataka koji se sastoji od centriranih slika lica ljudi zajedno sa ivicama dobijenih sa ovih slika, tako da se one koriste kao konture odnosno nacrti lica ljudi. Samim tim se od generativnog modela očekuje da nakon obučavanja bude sposoban da na osnovu proizvoljne skice osobe generiše njenu sliku.

Arhitekture generativne i diskriminatore mreže, kao i sama ideja za uslovno treniranje GAN-a je preuzeta iz rada popularno poznatog pod imenom pix2pix. Prva od dve glavne specifičnosti ovog rada je upotreba enkoder-dekoder mreže za translaciju slika za razliku od klasičnog pristupa semplovanja latentne promenljive *z* iz normalne ili uniformne distribucije. Druga specifičnost je upotreba kondicionalnog odnosno uslovljenog diskriminatora koji ne procenjuje samo da li je slika prava ili lažna na osnovu same slike, već ujedno ima pristup i uslovnoj slici na osnovu koje generator kreira svoju sliku ili koja je povezana sa postojećom pravom slikom, na taj način se postiže da diskriminator ujedno vrši procenu da li generisana slika odgovara zadatom uslovu. Pristup rada ovog GAN-a je takav da skup podataka mora da sadrži parove slika iz domena koji služi kao uslov i slika iz ciljnog domena.

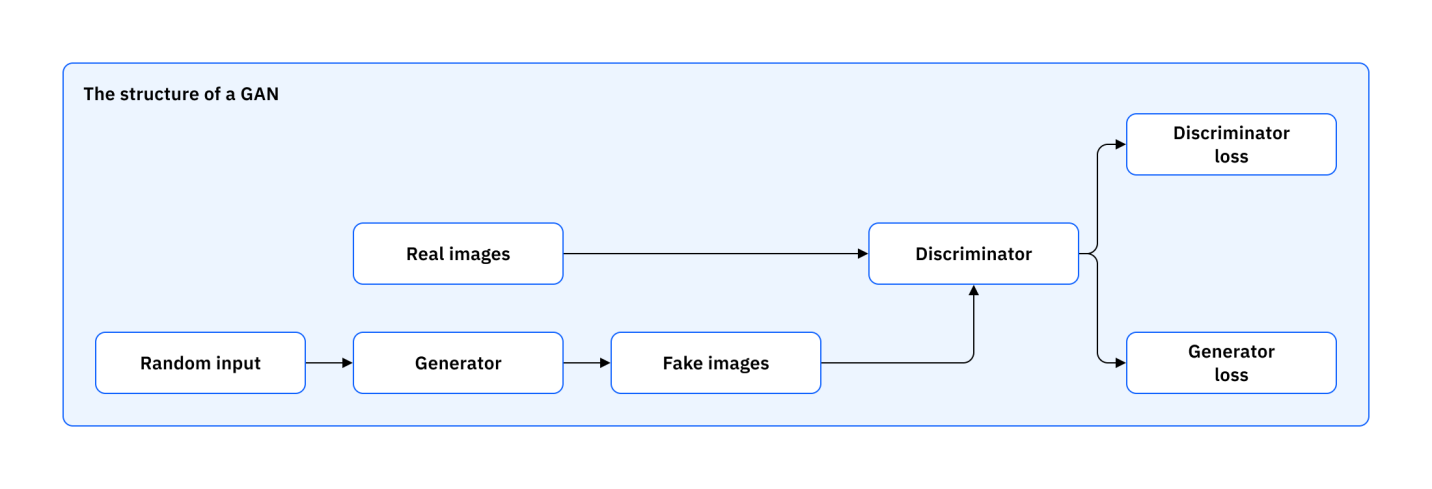
Izveštaj je dat u sledećem formatu. U drugom poglavlju je dat teorijski pregled generativnih suparničkih mreža i uslovljene translacije slika. Zatim, treće poglavlje predstavlja kraći osvrt na korišćeni skup podataka. Nakon toga, u četvrtom poglavlju su prikazani detalji implementacije obučavane mreže. U okviru petog poglavlja je prikazan proces i ishod, odnosno, rezultat treninga.

## **TEORIJSKI PREGLED**

Generativne suparničke mreže (generative adversarial network – GAN) su model mašinskog učenja koji generiše realne podatke učenjem obrazaca sa postojećih skupova podataka. Spada u tehnike nenadgledanog učenja kod dubokog učenja. Konkretno kod GAN-ova dve neuronske mreže rade u opoziciji jedna drugoj, jedna generiše podatke, dok pri tome druga vrši evaluaciju podataka tako da daje svoj sud o tome da li su podaci pravi ili generisani. Duboko učenje je već poznato po tome što daje dobre rezultate u zadacima koji se zasnivaju na izvlačenju infomacija sa slika, uključujući klasifikaciju slika, detekciju objekata i segmentaciju. Međutim, generisanje novih, realističnih slika predstavlja veći izazov zbog kompleksnosti izračunavanja generativnih modela.

GAN-ovi kao pojam su prvi put uvedeni od strane Ian Goodfellow-a u njegovom radu 2014. godine i predstavljaju ključno otkriće za izazov generisanja podataka. Iako su danas aktuelni i drugi pristupi koji daju i bolje rezultate od GAN-ova, poput difuzionih modela i generativnih modela zasnovanih na takozvanoj transformer arhitekturi, GAN-ovi i dalje predstavljaju popularnu i često korišćenu arhitekturu sa mogućnostima generisanja realističnih podataka.

GAN arhitektura se sastoji od dveju dubokih neuronskih mreže: generatora i diskriminatora. Obučavanje GAN-ova je proces koji uključuje kreaciju sintetičkih podataka od strane generatora na osnovu nasumičnog ulaza, to jest šuma, koji mu se zadaje. Diskriminator vrši evaluaciju generisanih uzoraka i stvarnih uzoraka iz skupa podataka nad kojim se vrši obučavanje, tako da on pridružuje skor između nula i jedan svakoj od slika. Skor jedan znači da slika izgleda realno a skor 0 znači da izgleda lažno. Backpropagation se zatim koristi kako bi se izvršila optimizacija obeju mreža. Dakle, generator koristi povratnu informaciju od diskriminatora kako bi napredovao i uspeo da kreira realističnije podatke. Na slici 2.1 je data tipična struktura GAN sistema.



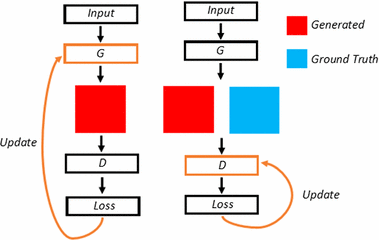
Slika 2.1. Tipična struktura GAN sistema.

Obučavanje GAN arhitekture predstavlja suparnički proces. Generator pokušava da prevari diskriminator i navede ga da klasifikuje sintetičke podatke kao prave, dok diskriminator konstantno unapređuje svoju sposobnost da razlikuje sintetičke i prave podatke. Ovaj proces je vođen funkcijom greške koja određuje performanse svake od mreža. Greška generatora je mera koliko dobro generator može da prevari diskriminator kako bi on verovao da su podaci generatora stvarni. Greška kod diskriminatora predstavlja meru koliko dobro diskriminator može da razlikuje veštačke i prave podatke. Dakle, niska greška diskriminatora znači da diskriminator uspešno identifikuje veštačke podatke. Pri tome, empirijski je zaključeno da ako diskriminator isuviše dobro identifikuje veštačke podatke, odnosno ako se vrednost njegove greške značajno približi nuli, generator neće biti u stanju da napreduje u toku obučavanja. Samim tim, izuzetno je važno da se obezbedi da diskriminator ne napreduje mnogo brže od generatora, što je česta pojava jer on ima relativno lakši zadatak od generatora.

Na primer, u toku obučavanja GAN-a na slikama pasa, generator transformiše nasumični šum koji dobija kao ulaz u slike koje liče na pse, dok diskriminator vrši evaluaciju tih slika naspram stvarnih slika pasa iz skupa trening podataka.

Vremenom ovakav suparnički proces dovodi do napredovanja obeju ovih mreža. To jest omogućava generatoru da nauči da kreira realistične ubedljive slike koje su bliske originalnom skupu podataka nad kojim je vršen trening. Za to vreme diskriminator usavršava svoju sposobnost da identifikuje suptilne razlike između pravih i generisanih slika.

U toku obučavanja GAN-a naizmenično se vrše ažuriranja težina generatora i diskriminatora. Diskriminator se obučava gradijentnim spustom i to tako što se za veštačke slike ažuriraju težine diskriminatora tako da on daje izlaz što bliže nuli dok se za prave slike vrši ažuriranje težina tako da izlaz bude što bliži jedinici. Ovo je moguće učiniti, takođe, pomoću gradijenta i to korišćenjem gradijentnog porasta umesto spusta. Pokazano je da korišćenje gradijentnog porasta daje bolje rezultate nego gradijentni spust vrednosti razlike jedinice i ocene diskriminatora. Sa druge strane generator se obučava tako što se na osnovu nasumičnog šuma kreiraju slike i onda se dobija ocena tih slika od strane diskriminatora, nakon čega se pomoću backpropagation algoritma ažuriraju samo parametri generatora tako izlaz generatora bude što bliži jedinici. Dakle, ne menjaju se težine diskriminatora jer nije cilj da se pogoršaju njegove performanse već da samo generator nauči da bolje prevari diskriminator. Ovakav način obučavanja je grafički ilustrovan na slici 2.2.



Slika 2.2. Obučavanje GAN-a.

Diskriminator praktično predstavlja funkciju greške generatora. Takav pogled na GAN daje i objašnjenje motivacije za uvođenje diskriminatora. Naime, za zadatke poput generisanje realističnih slika ne postoji matematička formula funkcije greške. Odnosno, ne postoji formula po kojoj bi se moglo odrediti da li je slika realna ili ne. Upravo se iz tog razloga koristi model koji će služiti kao funkcije greške. Gde pri tome nije moguće imati unapred istreniran model jer nije moguće unapred dobiti slike koje će generator kreirati u različitim fazama treninga a i dodatno kao što je već i pomenuto ako je diskriminator isuviše dobar onda brzina obučavanja generatora teži nuli.

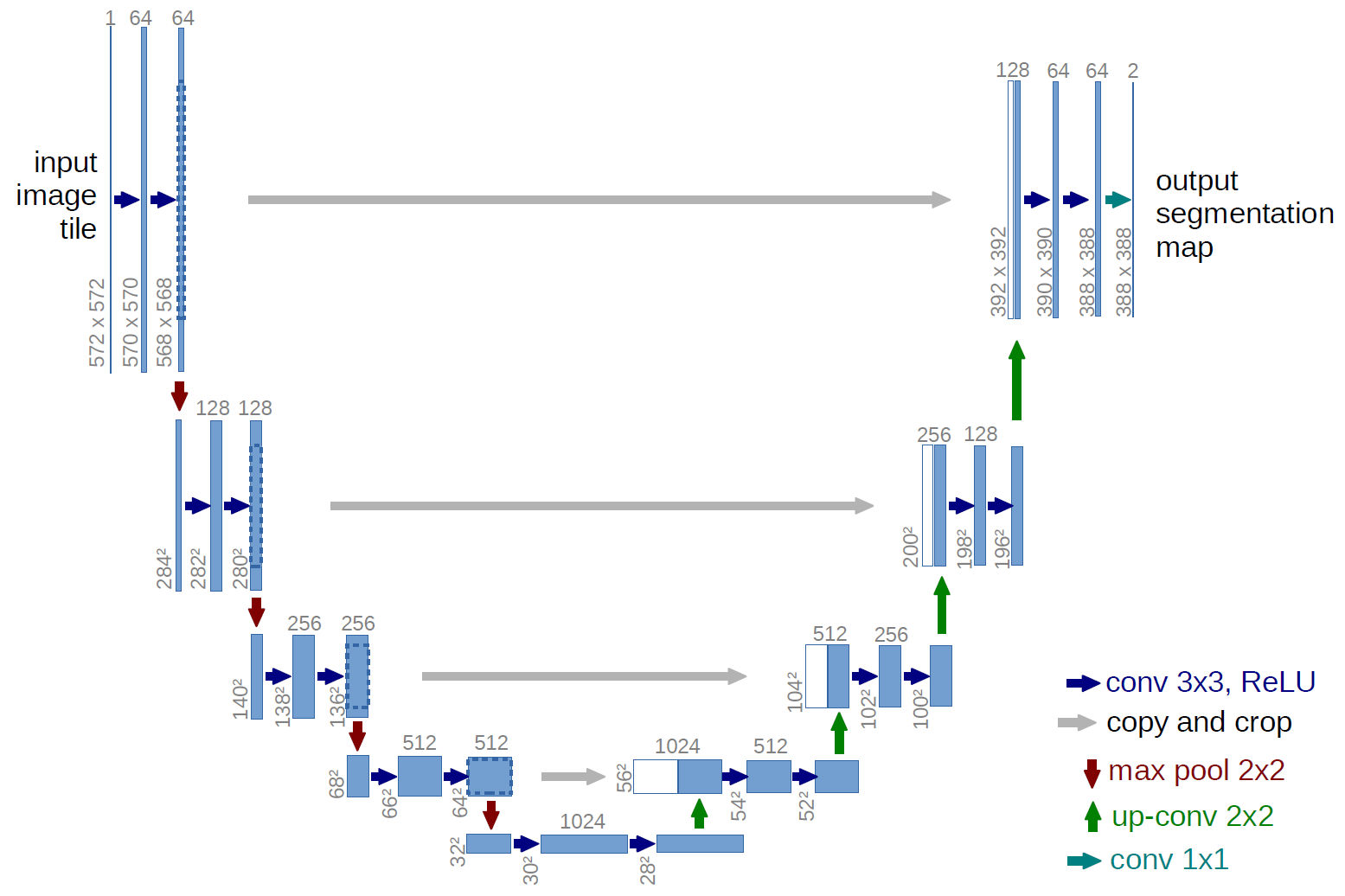
GAN-ovi se koriste za različite primene u računarskom vidu, generisanju slika, detekciji objekata, translaciji slika, generisanju slika na osnovu teksta, predviđanju sledećeg frejma u snimku itd. Samim tim, postoje različiti tipovi GAN-ova razvijani za konkretne potrebe:

* Vanila GAN: osnovna forma generativnih suparničkih mreža koja se sastoji od generatora i diskriminatora u suparničkoj igri. Vanila GAN-ovi se odlikuju korišćenjem jednostavnih MLP mreža za generatore i za diskriminatore. Jednostavni su za implementaciju ali su poznato nestabilni u toku obučavanja te je izuzetno teško sa njima postići dobre rezultate.
* Uslovni (conditional) GAN: označavaju se još i kao cGAN-ovi. Kod ovih GAN-ova obučavanje se vrši uz dodatne informacije koje se nazivaju oznake (labels) ili uslovi (conditions) i koje se prosleđuju i generatoru i diskriminatoru. Ove oznake pružaju kontekst i uz nasumični šum služe da omoguće generatoru da kreira podatke sa konkretnim zadatim karakteristikama. Ovaj tehnički izveštaj je vezan za implementaciju jednog ovakvog GAN-a. Pored ovde prikazane translacije slika, uslovni GAN-ovi se mogu koristiti na primer i za prevođenje crno-belih slika u obojene ali i za uslovljeno kreiranje na osnovu drugačijih tipova oznaka poput teksta. Dakle, moguće je kreirati GAN koji kreira sliku na osnovu teksta.
* Duboki konvolucioni GAN-ovi: predstavljaju grupu GAN-ova koja koristi konvolucione neuronske mreže i za generatore i za diskriminatore. Ovaj tip GAN-ova generiše značajno bolje rezultate u odnosu na vanila GAN-ove. Međutim, treba naglasiti da ovaj tip nije ortogonalan na tip uslovnih GAN-ova niti WGAN-ova i CycleGAN-ova koji će biti pomenuti u nastavku. Dakle, svi GAN-ovi koji koriste konvolucione neuronske mreže spadaju u ovaj tip GAN-ova a pri tome mogu spadati i u druge tipove.
* WGAN: grupa GAN-ova koji koriste Wasserstein funkciju greške. Kod ovih GAN-ova se ne koristi diskriminator već neuronska mreža sa sličnom svrhom koja se naziva kritičar. Glavna razlika u odnosu na ostale GAN-ove je to što kritičar ne vrši evaluaciju da li je slika realna ili ne već daje ocenu slike tako da veća vrednost ocene znači veći nivo realnosti. Pri tome ne postoji ograničenje niti raspon vrednosti iz koje ocene mogu biti izabrane. Na primer, moguća situacija je da generisana slika dobije ocenu 8 dok prava slika dobija ocenu 17, ili čak jedna ili obe ocene mogu biti negativne. Upravo je funkcija greške, kod koje se kritičar obučava da daje što veću ocenu realnim slikama a što manju ocenu generisanim, prethodno pomenuta Wasserstein funkcija.
* CycleGAN: odnosi se na konkretnu arhitekturu koja vrši prevođenje slika iz jednog domena u drugi domen bez potrebe korišćenja odgovarajućih ekvivalentnih parova slika iz tih domena. Konkretan primer dat u originalnom radu je bio vezan za prevođenje slika konja u slike zebra i obrnuto. Način na koji je ovo postignuto je bio upotrebom funkcije greške ciklične konzistentnosti. Naime, ako je jedan generator u stanju da prevede sliku iz domena A u domen B a drugi generator u stanju da prevede sliku iz domena B u domen A, onda uzastopni poziv ovih generator bi trebalo da kreira sliku identičnu polaznoj slici. Na taj način je moguće izračunati grešku ovakvog cikličnog procesa računanjem L1 ili L2 greške između polazne slike i slike dobijene upotrebom oba ova generatora-a. U ovu grupu GAN-ova spadaju i naredni radovi koji su ovakav pristup unapredili poput UNIT, MUNIT i DRIT GAN-ova, to jest svrstavaju se svi GAN-ovi koji koriste grešku ciklične konzistencije.
* Laplacian pyramid GAN (LAPGAN): GAN-ovi koji generišu slike visokog kvaliteta poboljšanjem istih na više različitih skala. Dakle, počinju tako što se generišu slike niske rezolucije i onda se postepeno dodaje veći broj detalja na višim rezolucijama korišćenjem niza GAN-ova. Ovaj pristup na većem broju skala je poznat kao Laplasova piramida.

### Uslovni GAN - pix2pix

Uslovni GAN-ovi generišu podatke tako što se generatoru zadaje konkretna kategorija odnosno tip podatka koji treba da generiše. Opciono je moguće zadati još i nasumični šum kako bi se postigao određeni nivo raznovrsnosti generisanih podataka. Zadata kategorija može biti na primer broj koji treba generisati prilikom treniranja na MNIST skupu podataka.

Pored toga moguće je zadati kompletnu sliku kao uslov, u tom slučaju ovaj proces podrazumeva translaciju zadate slike u ciljnu sliku. Generator tipično ima dekodersku strukturu, međutim u ovom slučaju takva struktura nije veoma zgodna pre svega zato što ona podrazumeva povećanje prostorne dimenzije po ceni smanjenje broja kanala fičer mapa. Međutim, kod translacije slika prostorna dimenzija ulaznog podataka odgovara uglavnom dimenziji očekivanog izlaza. Iz tog razloga se kod pix2pix i ostalih sličnih arhitektura koriste mreže koje se sastoje iz enkodera i dekodera.



Slika 2.3. U-Net arhitektura.

Endoker predstavlja deo mreže koji povećava broj kanala po ceni smanjenja prostorne dimenzije i njegova uloga je izvlačenje fičera odnosno obeležja sa slike. Nakon toga dekoder postepeno obnavlja prostornu dimenziju dok se ne dostigne potrebna veličina slike. U radu u kome je implementirana pix2pix arhitektura je konkretno generator implementiran kao arhitektura zasnovana na U-Net enkoder-dekoder arhitekturi. U-Net je poznat po tome što uvodi i dodatne veze između slojeva enkodera i dekodera koji imaju iste prostorne dimenzije. Konkretno vrši se konkatenacija fičer mapa iz enkodera na fičer mape dekodera. U-Net arhitektura je prikazana na slici 2.3. U četvrtom poglavlju će biti data i konkretna implementacija generatora.



Slika 2.4. Analiza rezultata dobijenih upotrebom različitih funkcija grešaka.

Za razliku od tradicionalnih GAN-ova za zadati ulaz je ovde poznat očekivani tačni izlaz koji generator treba da kreira. Tako da je moguća jednostavna upotreba L1 ili L2 funkcije greške za određivanje odstupanja kreirane slike i tačne slike. Međutim, u radu je empirijski prikazano da upotreba samo ovih funkcija greške dovodi do kreiranja mutnih slika. Zato je potrebno dodatno koristiti i klasičnu suparničku grešku koja predstavlja procenu odstupanja generisane slike od realne slike od strane diskriminatora. Pri tome diskriminator dodatno na ulazu dobija i informaciju o slici koja je korišćena kao ulaz u generator, na taj način on vrši procenu realnosti ali i procenu da li je slika odgovarajuća. Poređenje rezultata treninga korišćenjem samo L1 greške, samo suparničke greške i kombinacije ovih grešaka je prikazano na slici 2.4 koja je preuzeta iz rada.



Slika 2.5. Analiza rezultata za različiti broj patch-eva diskriminatora.

Dakle u okviru funkcije greške L1 greška se koristi kako bi generator naučio niske frekvencije te je dovoljno suparničkom greškom postići učenje visokih frekvencija. Kako bi model naučio visoke frekvencije dovoljno je da se diskriminator bavi samo strukturama u lokalnim delovima slike. Zato je u radu diskriminator projektovan tako ne daje globalnu ocenu cele slike već da daje pojedinačne ocene za veći broj manjih delova odnosno patch-ova slike. Samim tim diskriminator deli sliku na *N x N* nezavisnih oblasti. Empirijski je u radu pokazano da se najbolji rezultati postižu za relativno veliko *N* u odnosu na dimenzije slike. To jest konkretno, najbolji rezultati su prijavljeni za *N=70*, gde je pri tome dimenzija slike 256x256. Ovi rezultati, direktno preuzeti iz rada su prikazani na slici 2.5.

Matematički gledano ako je suparnička greška uslovnog GAN-a data izrazom:

, (1)

dok je L1 greška data izrazom :

, (2)

tada je krajnji cilj generatora dat izrazom:

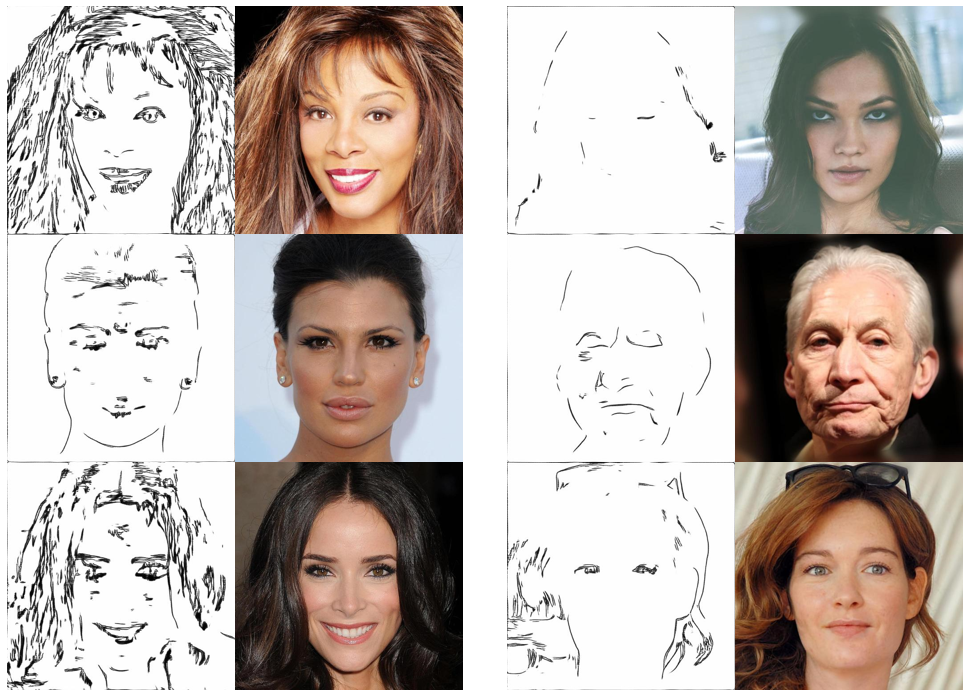
. (3)

Pri tome je u radu rečeno da se generator i diskriminator standardno treniraju naizmenično tako da se vrši jedan korak gradijentnog spusta na diskriminatoru pa jedan korak na generatoru. Dodatno izračunata greška diskriminatora se deli brojem dva kako bi se usporio stepen obučavanja diskriminatora u odnosu na generator. Za optimizaciju se preporučuje upotreba stohastičkog gradijentnog spusta uz Adam optimizator sa parametrima β1=0.5 i β2=0.999. Pri tome se koristi stopa učenja 0.0002. Dodatno naglašeno je da se Dropout i u toku testiranja i upotrebe mreže koristi kao i u toku treninga. Na taj način se postiže određeni stepen raznovrsnosti generisanih slika.

## **SKUP PODATAKA**

Skup podataka koji je korišćen za obučavanje GAN-a je javno dostupan na Kaggle-u, na linku: <https://www.kaggle.com/datasets/almightyj/person-face-sketches>. U pitanju su slike lica poznatih ličnosti uparene sa slikama kontura odnosno skica njihovih lica. Slike skupa podataka su preuzete iz CelebAMask-HQ skupa podataka, koji je javno dostupan na linku: <https://github.com/switchablenorms/CelebAMask-HQ>. To je skup slika lica poznatih ličnosti bez naočara i šešira. Slike su centrirane oko očiju i rotacija glava na slici oko vrata ja manja od dvadeset stepeni. Na slici 3.1 su dati primeri parova slika iz ovog skupa podataka, navedeni su primeri slika za koje su skice dobrog kvaliteta i slika kod kojih nije uhvaćen dovoljno veliki broj ivica kako bi skica bila upotrebiva, međutim svakako su sve slike bile korišćene za obučavanje modela.

U opisu skupa podataka navedeno je da su za ekstrakciju ivica slika, kojom se kreiraju skice lica korišćene operacije u Python programskom jeziku sledećim redosledom. Sve korišćene operacije su iz PIL biblioteke. Prvo je izvršena ekstrakcija ivica operacijom ImageFilter.FIND\_EDGES, zatim je slika zamućena operacijom ImageFilter.BLUR, nakon toga je slika ponovo izoštrena operacijom ImageFilter.SHARPEN nakon čega je invertovana kako bi se dobila bela slika sa crnim ivicama i to korišćenjem operacije ImageOps.invert() i naposletku je iskorišćena operacija ImageEnhance.Contrast().enhance(1.5). Nakon toga po uputstvima je potrebno iskoristiti model dostupan na linku: <https://esslab.jp/~ess/en/research/sketch/>, koji vrši pojednostavljenje skica zadržavajući najjače linije i koji slike prevodi u binarne slike.



Slika 3.1 Primeri iz korišćenog skupa podataka.

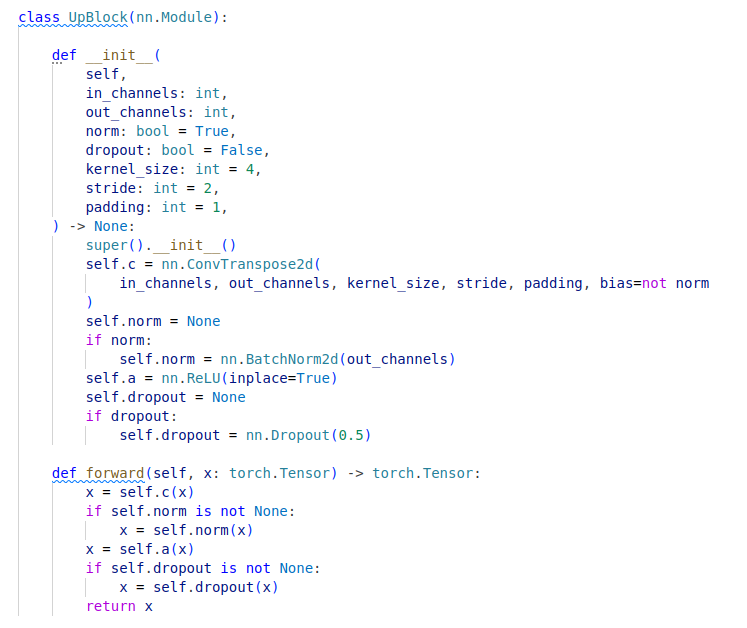
## **IMPLEMENTACIJA**

Model je implementiran korišćenjem PyTorch frejmvorka i to što je sličnije moguće modelu iz rada. Posebno su definisane i implementirane klase generatora i diskriminatora. Konkretno kako se generator sastoji iz enkodera i dekodera, tako da enkoder sadrži takozvane downsampling blokove koji smanjuju prostornu dimenziju povećavajući pri tome broj kanala, dok dekoder sadrži upsampling blokove koji povećavaju prostornu dimenziju smanjujući broj kanala, posebno su kreirane klase koje odgovaraju downsampling blokovima a posebno klase koje odgovaraju upsampling blokovima. Pri tome diskriminator takođe sadrži downsampling blokove, Klasa downsampling bloka je definisana tako da se mogu zadati vrednosti veličine jezgra, padding-a i stride-a prilikom instanciranja downsampling blokova.



Slika 4.1. Implementacija downsampling bloka.

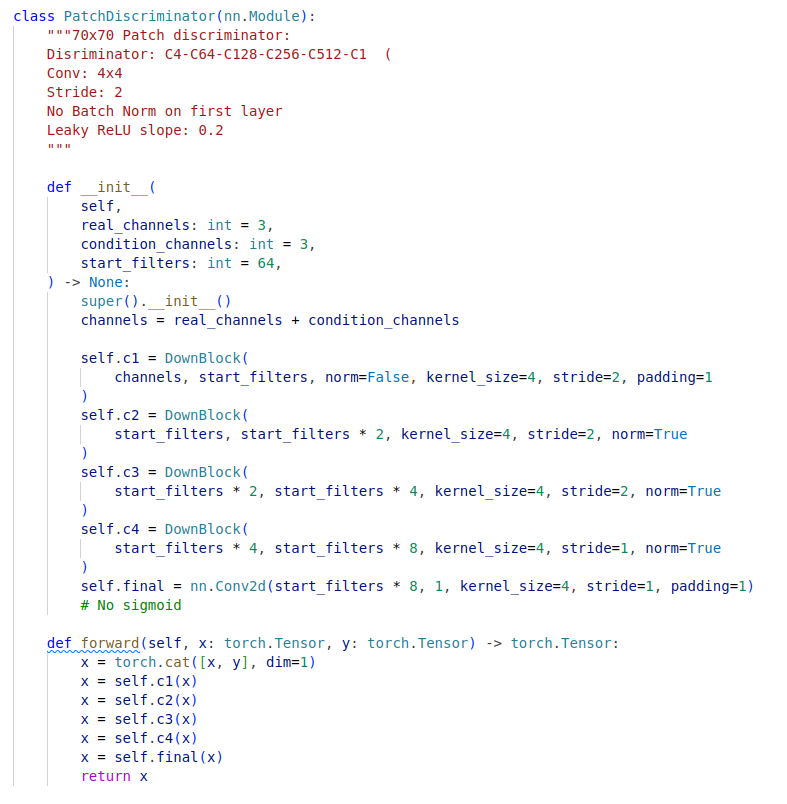
Na osnovu opisa datog u radu, downsampling blok se kreira tako da sadrži jedan konvolucioni sloj praćen opcionom Batch normalizacijom nakon koje se poziva LeakyReLU aktivacija. U radu se u svakom bloku ili koristi Batch normalizacija ili se ne koristi normalizacija, zato je downsampling blok implementiran tako da samo pruža izbor da li će ovaj tip normalizacije biti korišćen ili ne, dok se ne dozvoljava izbor različitog tipa normalizacije. LeakyReLU po instrukcijama u radu ima vrednost koeficijenta 0.2 za negativne vrednosti. Na slici 4.1 je prikazana implementacija downsampling bloka.



Slika 4.2. Implementacija upsampling bloka.

Upsampling blok funkcioniše slično kao i downsampling blok gde samo umesto konvolucije se koristi transpose konvolucija. Dodatak kod upsamplinga je to što se koristi i Dropout sa verovatnoćom isključenja aktivacija 0.5. Razlog za upotrebu Dropout u radu je postizanje različitosti izlaza za istu početnu sliku. Iz tog razloga se, suprotno tradicionalnom pristupu, Dropout i u toku testiranja koristi kao i u toku treninga, odnosno nema isključivanja Dropout-a za upotrebu modela. Poslednja razlika u odnosu na upsampling je to što se ne koristi LeakyReLU aktivacija nego klasičan ReLU. Na slici 4.2 je prikazana implementacija upsampling bloka.

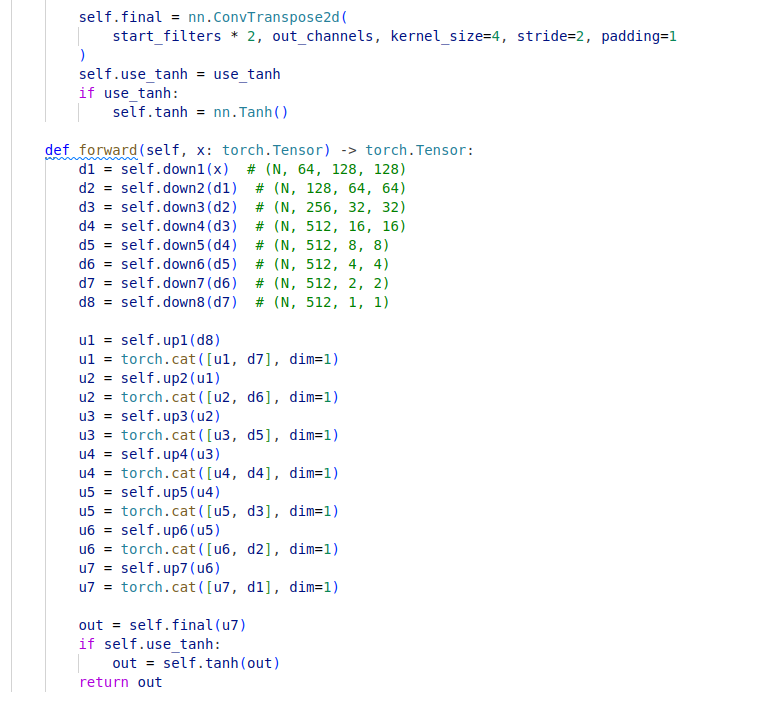
U radu je diskutovano par različitih arhitektura generatora i diskriminatora u zavisnosti od veličine slika na kojima se mreža obučava. Ovde je konkretno implementiran diskriminator koji se sastoji od 4 downsampling bloka i jednog sloja konvolucije. Pri tome, kao što je naglašeno u radu, prvi downsampling blok ne koristi normalizaciju dok ostala tri koriste. Stride konvolucije u prva tri bloka je 2 dok je u poslednjem bloku i zadnjem sloju konvolucije 1. U svakom bloku i poslednjoj konvoluciji padding je stavljen na 1. Na ovaj način se na kraju upravo dobija patch diskriminator koji deli sliku na 70x70 segmenta. Empirijski je u radu pokazano da ovakav diskriminator daje najbolje rezultate. Implementacija diskriminatora je prikazana na slici 4.3.



Slika 4.3. Implementacija diskriminatora.

Generator je implementiran tako da se sastoji iz osam downsampling blokova, sedam upsampling blokova i poslednje transpose konovlucije na koju se primenjuje tangens hiperbolički. Prvi i poslednji downsampling blokovi ne sadrže normalizaciju dok je ostali blokovi sadrže. Svaki downsampling blok sadrži jezgro veličine 4x4, padding 1 i stride 2. Početni broj kanala ulazne slike je tri, dok prvi blok sadrži šezdeset i četiri kanala. Drugi, treći i četvrti blokovi dupliraju broj kanala dok peti, šesti, sedmi i osmi blokovi imaju konstantan broj kanala odnosno petsto dvanaest kanala. Nakon toga svih sedam upsampling blokova sadrže normalizaciju a samo prva tri i Dropout, dok im je svima jezgro veličine 4x4, stride 2 i padding 1. Kako je generator baziran na U-Net arhitekturi kod njega treba uzeti u obzir povećanje broja kanala zbog konkatenacije slojeva, tako da se brojevi kanala po slojevima dekodera menjaju na sledeći način: prvi upsampling blok sadrži petsto dvanaest kanala drugi kao i treći i četvrti takođe sadrže petsto dvanaest, dok peti sadrži dvesta pedeset i šest, šesti sadrži sto dvadeset osam kanala i poslednji sedmi sadrži šezdeset i četiri. Na slici 4.4 je prikazana implementacija generatora.



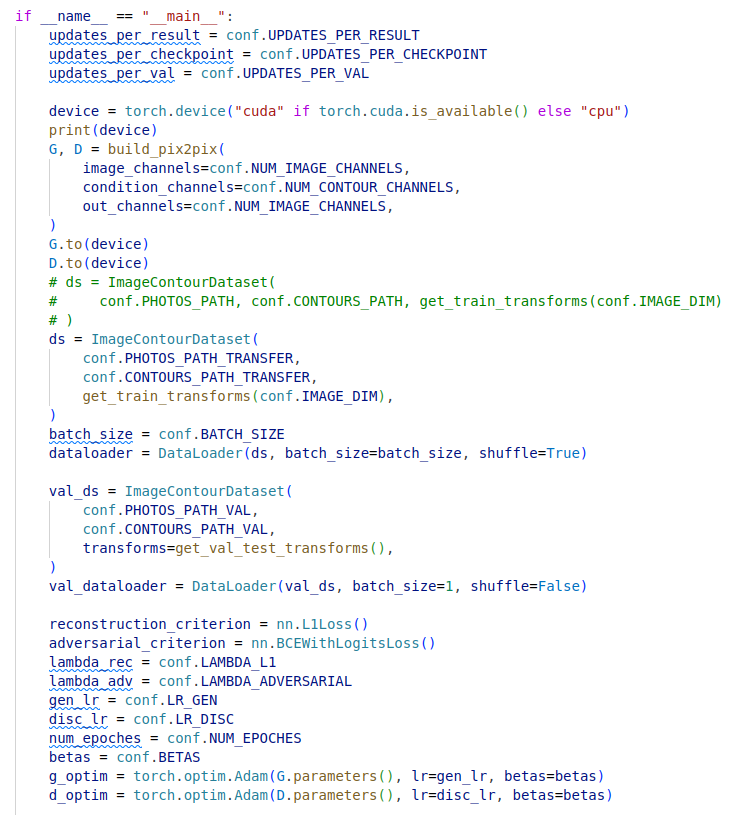


Slika 4.4. Implementacija generatora.

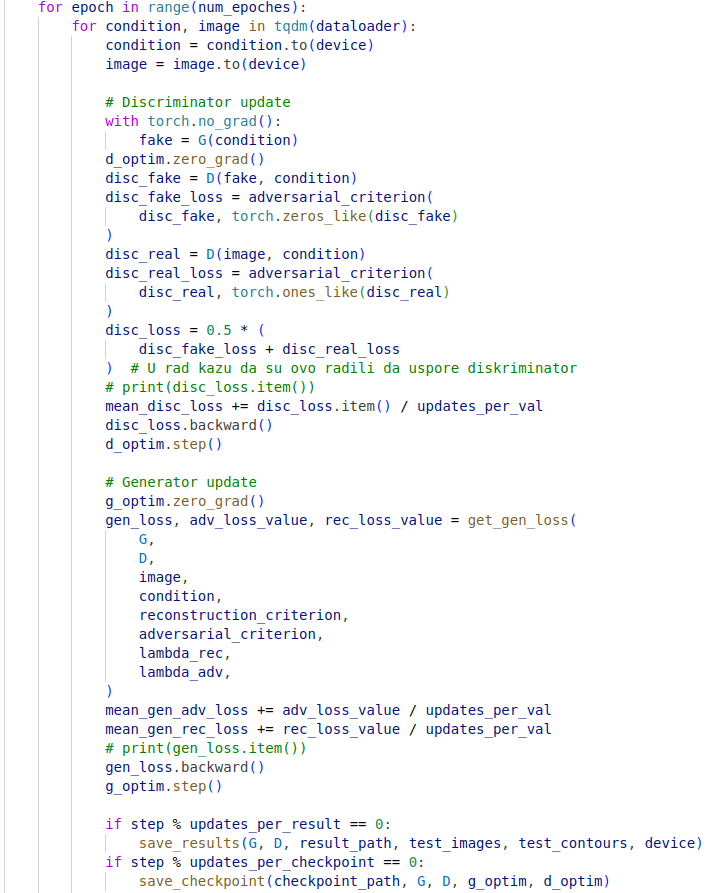
Trening petlja je klasične strukture, prvo se vrši inicijalizacija modela generatora i diskriminatora, nakon toga se inicijalizuje i objekat klase za predstavljanje skupa podataka, klasa se naziva ImageContourDataset i njena implementacija je data na slici 4.5. Nakon toga se pomoću objekta skupa podataka inicijalizuje dataloader sa batch veličinom od 32 slike. Za optimizaciju i generatora i diskriminatora se koriste posebno inicijalizovani objekti Adam optimizatora. Prema preporuci iz rada vrednosti beta prametara su (0.5, 0.999). Kao što je prethodno pominjano za izračunavanje greške se koristi kombinacija L1 greške i suparničke greške koja se određuje pomoću funkcije BCEWithLogitsLoss(). Nakon što se sve prethodno pomenuti objekti inicijalizuju, što je prikazano na slici 4.6, kreće izvršenje same petlje u okviru koje se za svaku epohu iz zadatog broja epoha, za svaki batch iz skupa podataka, vrši jedan korak obučavanja diskriminatora praćen korakom obučavanja generatora, kao što je prikazano na slici 4.7. Ujedno, u okviru ove petlje vodi se računa o broju ažuriranja težina modela. To se radi zato što postoje zadati parametri broja izmena nakon kojeg se snimaju rezultati sa trening skupa, rezultati sa validacionog skupa i grafikon funkcije greške kroz vreme.



Slika 4.5. ImageContourDataset.



Slika 4.6. Inicijalizacija objekata trening petlje.

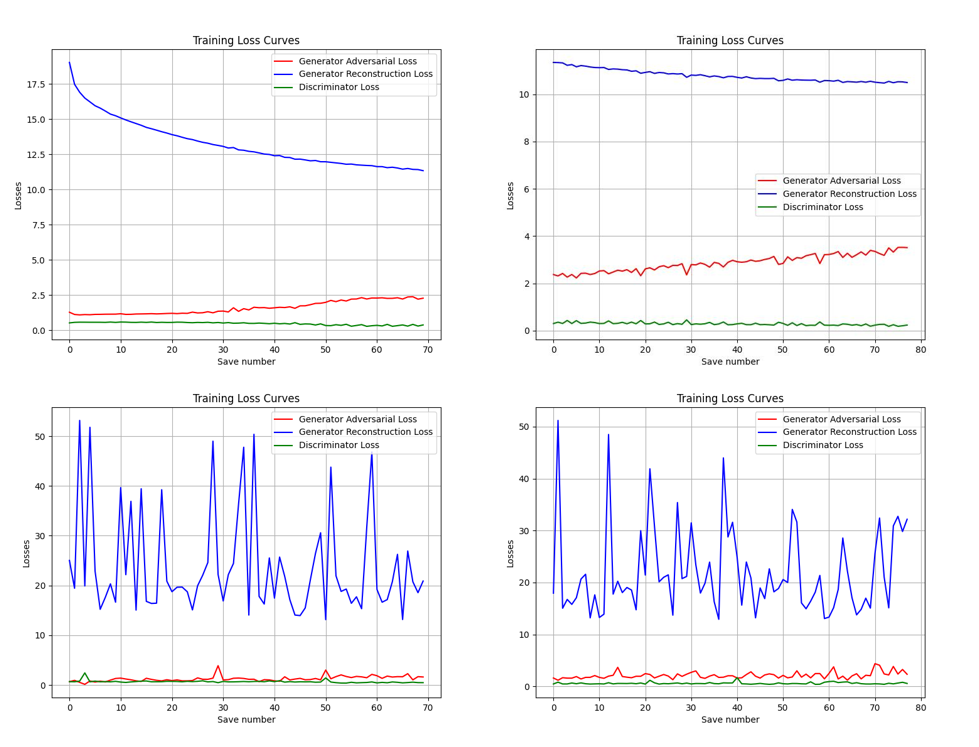


Slika 4.7. Implementacija trening petlje.

## **REZULTATI**

Obučavan je veći broj modela tako da je isprobano kakvi rezultati će se dobiti za različite vrednosti koeficijenta kojim se množi rekonstruktivna L1 greška. U radu na osnovu koga je implementiran ovaj model je preporučena upotreba koeficijenta sa vrednošću 100. Trenirani su modeli i sa značajno manjom vrednošću ovog koeficijenta ali su ipak najbolji rezultati dobijeni kada je praćena preporuka iz rada te će samo ti rezultati biti prikazani u nastavku. Trening skup podataka sadrži 20 655 slika, odnosno kada se koriste 32 slike po batch-u to znači da se jedna epoha sastoji od 646 izmena. Rezultati i funkcije greške su praćene na svakih 1000 izmena, dakle nešto manje od svake dve epohe. Izvršena je ukupno 148 snimanja rezultata u toku treninga modela.

Obučavanje je izvršeno iz dva dela, naime usled određenih okolnosti bilo je neophodno napraviti pauzu u toku obučavanja. Dakle, drugi deo obučavanja je nastavljen jednostavno učitavanjem vrednosti težina i vrednosti optimizatora poslednjeg snimljenog modela. Na slici 5.1 u gornjem redu su grafički prikazane komponente funkcije greške za trening skup u provj fazi (levo) i u drugoj fazi (desno) dok su u donjem redu prikazane ove komponente za validacioni skup podataka.



Slika 5.1. Funkcije greške u toku obučavanja.

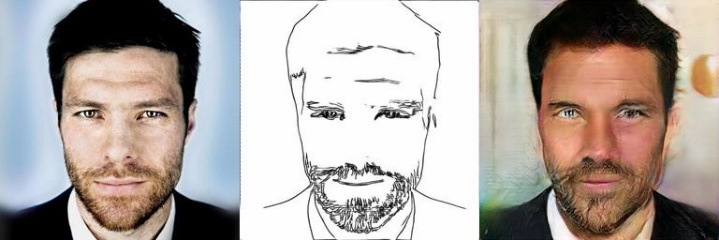
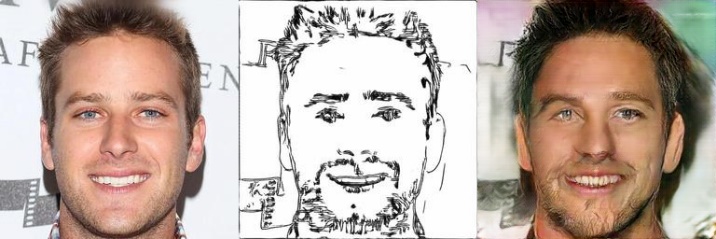
Jasno se vidi opadanje L1 komponente funkcije greške vremenom na trening skupu. Ova funkcija nam jedina omogućava uporedivo praćenje performansi modela. Naime, kako se i diskriminator menja u toku obučavanja GAN-a to znači da greška generatora u jednoj epohi nije uporediva sa greškom generatora u nekoj narednoj epohi jer se je i sam diskriminator koji praktično predstavlja funkciju greške izmenio. Međutim, prikaz suparničkih grešaka je i dalje koristan kako bi bilo moguće proveriti da diskriminator nije isuviše uznapredovao u odnosu na generator jer bi to onemogućilo nastavak obučavanja.

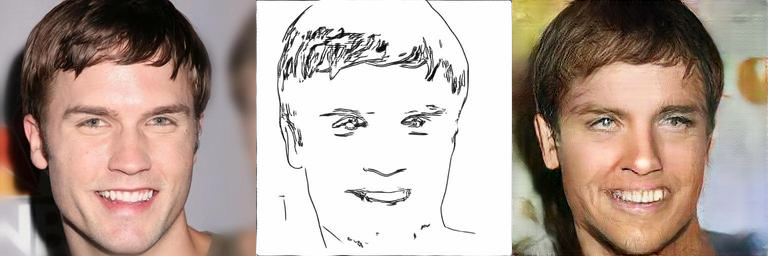


Slika 5.2. Primer slike sa dobrom vrednošću za L1 grešku koja je vizuelno loša.

L1 greška na validacionom skupu je skokovita što obično znači da model ne uspeva da nauči korisne informacije u toku obučavanja, što bi značilo da trenuci kada je ova greška niska na validacionom skupu predstavljaju situacije gde model skoro slučajno daje dobre performanse na validacionom skupu te samim tim on neće davati dobre performanse i na test skupu. Međutim, uprkos neskokovitosti L1 greške na trening skupu podataka, vizuelnom inspekcijom rezultata sa trening skupa, za slučajeve kada se na validacionom skupu dobijaju loše vrednosti L1 greške, primećeno je da se u ovim trenucima takođe javljaju i loše slike u trening skupu podataka i to slike koje su prekomerno blede. Na osnovu toga je donesen zaključak da verovatno postoji određeni nivo odstupanja raspodele podataka u trening i validacionom skupu, to jest konkretno pretpostavka je da trening skupu sadrži veći broj slika sa belom pozadinom tako da generisane slike koje su većinski bele daju relativno dobar rezultat za L1 funkciju greške i pretpostavka je da to nije slučaj sa slikama u validacionom skupu podataka. Primer ovakve izbeljene slike na trening skupu je dat na slici 5.2. Na osnovu prethodnog zaključka sledi da je model upotrebiv uprkos stepenastoj grešci na validacionom skupu podataka, ovaj zaključak je potvrđen primenom obučenog modela na hold-out test skup podataka. Na slici 5.3 su dati primeri upotrebe obučenog modela na podatke iz test skupa, pri tome su prikazani rezultati bez obzira na njihovu uspešnost ali samo za one slike čije su konture bile zadovoljavajućeg kvaliteta.







Slika 5.3. Rezultati primene modela na test skup podataka.

## **LITERATURA**

[1] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, <https://arxiv.org/abs/1611.07004>.

[2] Specijalizacija za GAN-ove sa Coursera-e:

<https://www.coursera.org/specializations/generative-adversarial-networks-gans>

[3] ,,What are generative adversarial networks (GANs)? “, dostupno na:

<https://www.ibm.com/think/topics/generative-adversarial-networks>, 3.9.2025.

[4] Goodfellow, Ian J., et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014), <https://doi.org/10.48550/arXiv.1406.2661>.

[5] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015), <https://doi.org/10.48550/arXiv.1511.06434>.

[6] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017, <https://doi.org/10.48550/arXiv.1701.07875>.

[7] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017, <https://doi.org/10.48550/arXiv.1703.10593>.

[8] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Cham: Springer international publishing, 2015, <https://doi.org/10.48550/arXiv.1505.04597>.

[9] „Person Face Sketches “, dostupno na:

<https://www.kaggle.com/datasets/almightyj/person-face-sketches>, 3.9.2025.

[10] ,,CelebAMask-HQ “, dostupno na:

<https://github.com/switchablenorms/CelebAMask-HQ>, 3.9.2025.