UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Nikolina Kuprešanin

PROBLEMI STABILNOG OBUČAVANJA GENERATIVNIH SUPARNIČKIH MREŽA

master rad

Beograd, 2023.

Mentor:

dr Mladen Nikolić, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Jovana Kovačević, docent
Univerzitet u Beogradu, Matematički fakultet

dr Mirjana Maljković, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Porodici

Naslov master rada: Problemi stabilnog obučavanja generativnih suparničkih mreža

Rezime: Generativne suparničke mreže predstavljaju model nenadgledanog učenja zasnovan na originalnoj ideji suprostavljenih neuronskih mreža. Sastoje se iz dve mreže sa različitim ciljevima, generatora i diskriminatora. Generator pokušava da na osnovu vektora slučajnih vrednosti generiše podatke iz raspodele zadate trening skupom. Diskriminator vrši klasifikaciju podataka na stvarne i lažne (generisane od strane generatora). Napredovanje jedne mreže dovodi do slabijih rezultata druge i obrnuto. Zato obučavanje generativnih suparničkih mreža predstavlja izazovan zadatak koji ne garantuje konvergenciju. Ukoliko se diskriminator obuči da uspešno razlikuje stvarne od generisanih instanci, njegova povratna informacija neće biti dovoljno informativna za generator. Sa druge strane, ukoliko se generator obuči da generiše verodostojne podatke, diskriminator neće imati šansu da napreduje. Tokom obučavanja može doći do situacije kada generator proizvodi mali broj instanci, koje diskriminator pogrešno klasifikuje. Na taj način model ostaje zaglavljen u lokalnom minimumu. Iako su generativne suparničke mreže poznate po uspešnim rezultatima u praksi, ovo su samo neki od problema koji mogu nastati tokom njihovog obučavanja. U radu su predstavljeni najčešći problemi i predložena rešenja koja vode stabilnijem procesu obučavanja.

Ključne reči: mašinsko učenje, generativni modeli, neuronske mreže, WGAN

Sadržaj

Glava 1

Uvod

Mašinsko učenje je grana veštačke inteligencije koja postoji dugi niz godina. Poslednjih decenija svedoci smo naglog razvoja mašinskog učenja i njegove primene u svakodnevnom životu, od prepoznavanja glasa na mobilnim telefonima do otkrivanja lažnih bankovnih transakcija. Posebno su interesantni generativni modeli mašinskog učenja, koji na osnovu datog skupa podataka modeluju zajedničku raspodelu promenljivih koje čine te podatke. Tako omogućavaju stvaranje novih instanci iz te raspodele podataka.

Generativne suparničke mreže (eng. *Generative adversarial networks*) (GAN) predstavljaju generativni model mašinskog učenja zasnovan na ideji suparničkog obučavanja dve neuronske mreže sa ciljem međusobnog poboljšavanja. Naizmeničnim treniranjem, generator i diskriminator uče da postanu što tačniji u svojim predviđanjima. Na osnovu povratne informacije od diskriminatora, generator uči da generiše nove podatke iz raspodele zadate trening skupom. Sa druge strane, diskriminator uči da klasifikuje podatke na stvarne i generisane. Uspešna klasifikacija podataka usmerava generator da proizvodi uverljivije podatke. Generator možemo posmatrati kao plagijatora koji želi da iskopira stil poznatog slikara, dok je diskriminator kritičar koji prepoznaje da li je umetničko delo stvarno ili plagijat. Što je kritičar bolji u prepoznavanju umetničkih dela, plagijator će se truditi da proizvede verodostojnije slike. Obrnuto, što je plagijator bolji u generisanju slika, kritičar će više truda ulagati u prepoznavanje umetničkih dela i vršiće bolju klasifikaciju. Njihovi suprostavljeni ciljevi dovode do međusobnog poboljšavanja.

Od prvog predstavljanja GAN modela 2014. godine u radu Iana Gudfelova [9] pa do danas, generativne suparničke mreže našle su raznovrsne primene. Novi verodostojni podaci, kakve ovi modeli generišu, mogu se iskoristiti prilikom rešavanja

problema nebalansiranih skupova podataka ili kao zamena za poverljive podatke tokom njihovog prikrivanja. Prevashodno se koriste za generisanje slika, ali se mogu primeniti i na druge vrste podataka kao što je zvuk. Inovativna ideja na kojoj GAN počiva omogućila je stvaranje slika visoke rezolucije. Ipak treniranje generativnih suparničkih mreža je veoma zahtevno u praksi. Generator može naučiti da generiše uvek iste instance, koje diskriminator prihvata kao stvarne. Tako se pokriva samo mali deo raspodele podataka. Naizmenično obučavanje suparničkih mreža ne garantuje konvergenciju i ponekad rezultuje potpuno neinterpretabilnim izlazima iz generatora. Nedostatak odgovarajućih metrika za evaluaciju onemogućava upoređivanje dobijenih GAN modela. Tehnike za rešavanje problema kao što su kolaps modela, problem konvergencije i problem nestajućih gradijenata, omogućile bi bolje rezultate i dalji razvoj generativnih suparničkih mreža.

Cilj ovog rada je da ukaže na potencijalne probleme koji mogu nastati tokom obučavanja generativnih suparničkih mreža i opiše tehnike za njihovo rešavanje. Rad se sastoji od nekoliko poglavlja. Nakon uvodnog poglavlja, u drugom poglavlju se uvode osnovni koncepti mašinskog učenja neophodni za razumevanje rada. U okviru trećeg poglavlja predstavljene su matematičke osnove i teorijski pregled generativnih suparničkih mreža, sa posebnim osvrtom na duboke konvolutivne generativne suparničke mreže i proces obučavanja GAN-ova. Četvrto poglavlje daje uvid u najčešće probleme koji mogu nastati tokom obučavanja, zajedno sa predloženim tehnikama za njihovo rešavanje. U petom poglavlju opsana je implementacija predloženih rešenja i dati su rezultati njihove eksperimentalne evaluacije. Na praktičnom primeru izvršeno je upoređivanje opisanih tehnika. Na kraju su izvedeni zaključci sa osvrtom na celokupan rad.

Glava 2

Osnovni pojmovi

2.1 Podele mašinskog učenja

Mašinsko učenje se bavi rešavanjem problema za koje je teško dati formalnu specifikaciju, ali je dostupna velika količina podataka koji na neki način ilustruju rešenja pojedinačnih instanci problema. Sve veća dostupnost raznih vrsta podataka dovodi do naglog porasta primene mašinskog učenja. Na osnovu prirode problema koji se posmatra, oblast mašinskog učenja može se podeliti na nadgledano učenje, nenadgledano učenje i učenje potkrepljivanjem.

Nadgledano učenje (eng. *supervised learning*) karakteriše postojanje informacije o tome šta je potrebno naučiti. Vrednost koju algoritam treba da predvidi naziva se ciljna promenljiva, dok ulazne podatke nazivamo atributima. Algoritmu koji uči prosleđuju se uređeni parovi atributa i ciljnih promenljivih. Na osnovu njih algoritam treba da ustanovi zavisnost ciljne promenljive od atributa i nauči da za nove ulaze proizvede očekivane izlaze.

Nenadgledano učenje (eng. *unsupervised learning*) se odlikuje nedostatkom vrednosti ciljnih promenljivih. Algoritam koji uči dobija jedino skup ulaznih podataka između kojih je potrebno uočiti zakonitosti i doneti zaključke. Dok kod nadgledanog učenja algoritmi uočavaju zakonitosti između ulaznih vrednosti i date ciljne promenljive, ovde to nije moguće. Obrasci koji se uče zadati su skupom ulaznih podataka.

Učenje potkrepljivanjem (eng. *reinforcement learning*) primenjuje se kod problema do čijeg se rešenja dolazi nizom akcija. Tokom učenja nije poznato koja akcija je dobra u kojoj situaciji i dostupna je jedino informacija o krajnjem ishodu. Cilj učenja je da se na osnovu pokušaja i grešaka iz iskustva nauči koje akcije su dobre

u kojim situacijama.

Generativne suparničke mreže kao ulaz dobijaju skup podataka bez ciljnih promenljivih i samim tim predstavljaju model nenadgledanog učenja. Od generatora se očekuje da uoči obrasce između datih podataka i generiše nove. Dobro je napomenuti da je diskriminator u stvari jedan klasifikator podataka na stvarne i lažne i samim tim predstavlja model nadgledanog učenja.

Posmatrajmo skup instanci X i skup obeležja Y. U zavisnosti od toga koliku količinu podataka opisuju, modele mašinskog učenja možemo podeliti na generativne i diskriminativne.

Diskriminativni modeli opisuju zavisnost ciljne promenljive od atributa, odnosno modeluju uslovnu verovatnoću $p(y|x)$. Diskriminativni modeli pokušavaju da povuku granice između podataka, bez tačnog modelovanja čitave raspodele. Uslovna raspodela $p(y|x)$ je često jednostavnija, samim tim i lakša za naučiti od zajedničke raspodele $p(x, y)$ [16].

Generativni modeli opisuju zajedničku raspodelu podataka, to jest modeluju verovatnoću $p(x, y)$. Generativni modeli uočavaju zavisnosti između svih promenljivih, ne samo zavisnost ciljne promenljive od atributa. Poznavanje zajedničke raspodele daje potpuni uvid u podatke i omogućava generisanje novih instanci iz te raspodele. Kako je potrebno pronaći zavisnosti između velikog broja promenljivih, generativni modeli zahtevaju veliku količinu podataka na osnovu koje će je uočiti. Generativne suparničke mreže predstavljaju vrstu generativnih modela.

2.2 Neuronske mreže

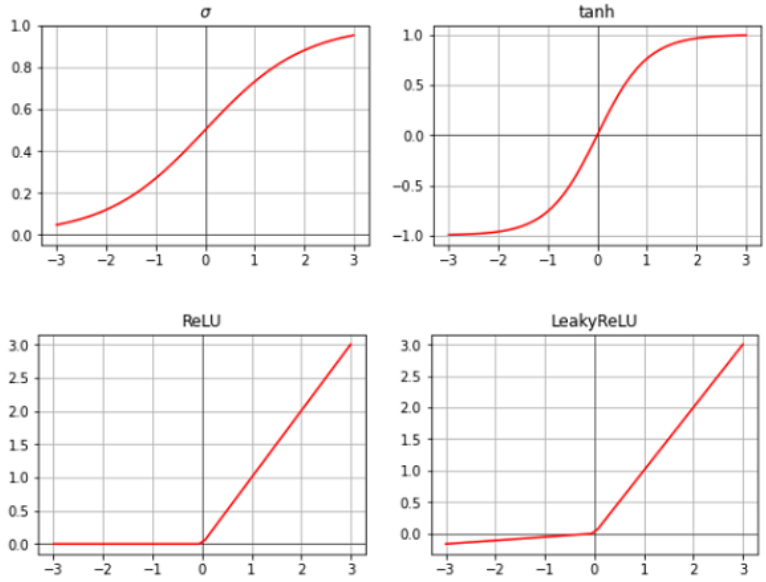
Ljudski cerebralni korteks sadrži oko 10^{10} nervnih ćelija zaduženih za obradu signala. Preko sinaptičkih veza svaki neuron se povezuje sa hiljadama drugih, obrazujući mrežu u cerebralnom korteksu zaduženu za obradu vizuelnih, audio i senzornih podataka [15]. Jedna od najpopularnijih metoda mašinskog učenja, neuronske mreže, inspirisana je strukturom ljudskog mozga.

Neuron je osnovna jedinica građe neuronskih mreža zadužena za obradu signala primljenih od susednih neurona. Sa stanovišta mašinskog učenja signali predstavljaju realne brojeve, a neuron je funkcija koja vrši nelinearnu transformaciju linearne kombinacije svojih ulaza. Nelinearnu transformaciju nazivamo aktivaciona funkcija i njen izbor zavisi od primene. Neke od najpoznatijih aktivacionih funkcija su sigmoidna, tangens hiperbolički, ispravljačka linearna jedinica (eng. $ReLU$) i nakošena

ispravljačka linearna jedinica (eng. *LeakyReLU*) i one će biti prisutne u ovom radu. Grafici funkcija dati su na slici 2.1.

$$\sigma(x) = \frac{1}{1+e^{-x}} \text{ , } tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \text{ , } ReLU(x) = max(0,x)$$

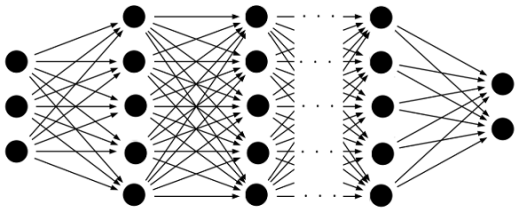
$$LeakyReLU(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$$



Slika 2.1: Grafici aktivacionih funkcija

Napomenimo da *LeakyReLU* predstavlja modifikaciju *ReLU* aktivacione funkcije koja za ulaz x manji od nula uzima vrednost αx, za malu vrednost parametra α.

Potpuno povezane neuronske mreže predstavljaju najstariji model neuronskih mreža. Mreža se sastoji od neurona grupisanih u slojeve. Svi neuroni jednog sloja kao ulaze primaju izlaze svih neurona prethodnog sloja, a svoje izlaze prosleđuju svim neuronima narednog sloja. Zbog specifične arhitekture su dobile naziv potpuno povezane mreže. Prvi sloj neurona naziva se ulazni i njegovi ulazi predstavljaju ulaze mreže. Poslednji sloj naziva se izlazni sloj i njegovi izlazi predstavljaju izlaze potpuno povezane mreže. Preostali slojevi nazivaju se skriveni slojevi i ukoliko postoje potencijalno ih može biti veoma mnogo. Neuronske mreže obično imaju makar jedan skriveni sloj, a ukoliko imaju više skrivenih slojeva mreže se nazivaju duboke neuronske mreže. Primer strukture potpuno povezane neuronske mreže nalazi se na slici 2.2.



Slika 2.2: Struktura potpuno povezane mreže

Neuroni ulaznog sloja primenjuju svoju aktivacionu funkciju na ulazne podatke i prosleđuju rezultate neuronima narednog sloja. Svi neuroni narednog sloja primenjuju svoje aktivacione funkcije na dobijene ulaze i prosleđuju svoje izlaze narednom sloju. Ponavljanjem postupka rezultati se propagiraju do izlaznog sloja. Opisani postupak dovodi do formalne definicije potpuno povezanih neuronskih mreža.

Definicija 2.1: *Potpuno povezana neuronska mreža je funkcija F_p definisana sledećim iterativnim postupkom*

$$h_0 = x$$
$$h_i = f(W_i h_{i-1} + b_i),\ i = 1, 2, ..., N$$
$$F_p = f_1(W_N h_{N-1} + b_N)$$

Gde x predstavlja vektor ulaznih podataka, N broj slojeva, f i f_1 aktivacione funkcije, b_i vektor slobodnih članova neurona i-tog sloja, W_i matricu parametara čije su vrste vektori parametara koji množe ulazne vrednosti neurona i-tog sloja. Vektori b_i i matrice W_i predstavljaju parametre neuronske mreže p koje je potrebno naučiti.

Primetimo da funkcija f_1 koja se primenjuje u poslednjem sloju može biti različita od funkcije f, u zavisnosti od primene. Tokom obučavanja potrebno je izvršiti izbor parametara p tako da je greška učenja minimalna. Ovo se postiže definisanjem odgovarajuće funkcije greške i njenom minimizacijom. Izbor funkcije greške L koja meri odstupanje stvarnih od očekivanih vrednosti zavisi od primene.

Ukoliko neuronska mreža F_p predstavlja klasifikator koji razvrstava podatke u jednu od datih n klasa, preporučena funkcija greške je funkcija *unakrsne entropije* zadata narednom formulom:

$$L(F_p(u), v) = -\sum_{i=1}^{n} v_i log F_p^{(i)}(u)$$

Sa $F_p^{(i)}$ označena je i-ta koordinata vektorske funkcije F_p.

Optimizacija neuronskih mreža

Minimizacija funkcije greške vrši se pretraživanjem prostora parametara u potrazi za odgovarajućom kombinacijom koja grešku najviše približava nuli. Optimizacija mreže vrši se gradijentnim metodama. Napomenimo da neuronske mreže nastaju kombinovanjem uglavnom diferencijabilnih funkcija, ali i nediferencijabilnih kao što je $ReLU$. Nediferencijabilnost u nuli $ReLU$ aktivacione funkcije može se rešiti dodefinisanjem gradijenta u nuli. Skup tačaka u kojima je funkcija nediferencijabilna je mere nula i ne predstavlja problem prilikom optimizacije mreže [14].

Gradijentni spust predstavlja iterativni optimizacijski postupak za pronalaženje lokalnog minimuma diferencijabilne funkcije. Gradijent je vektor parcijalnih izvoda funkcije i određuje smer najbržeg rasta funkcije. Kretanjem u smeru suprotnom od gradijenta traži se minimum funkcije. Polazeći od odabrane polazne tačke x_0 svaka naredna se dobija primenom pravila $x_{i+1} = x_i - \alpha_i \nabla f(x_i)$, gde α_i predstavlja dužinu i-tog koraka. Zbog kompleksnosti neuronskih mreža izračunavanje gradijenta nije trivijalan zadatak.

Algoritam propagacije unazad (eng. *backpropagation*) predstavlja algoritam za izračunavanje gradijenata korišćen pri treniranju neuronskih mreža. Kako se mreže sastoje od primene aktivacionih funkcija na linearne transformacije svojih ulaza potrebno je omogućiti efikasno izračunavanje izvoda složenih funkcija. U algoritmu se koristi pravilo izračunavanja parcijalnih izvoda kompozicije funkcija. Parcijalni izvod kompozicije funkcija $f : \mathbb{R}^n \to \mathbb{R}$ i $g : \mathbb{R}^m \to \mathbb{R}$ računa se na sledeći način:

$$\partial_i(f \circ g) = \sum_{j=1}^{n} (\partial_j f \circ g)\partial_i g_j$$

U praktičnoj primeni algoritam se sastoji iz dva prolaza kroz neuronsku mrežu. Pri prolazu unapred za zadati ulaz izračunavaju se izlazi mreže. Pri prolazu unazad računa se greška odstupanja izlaza i propagira se unazad. Krećući se unazad kroz slojeve, od poslednjeg do prvog, proširuju se do tada izračunati parcijalni izvodi.

2.3 Konvolutivne neuronske mreže

Poslednjih godina postignuti su značajni rezultati na polju računarskog vida. Posebnu zaslugu za to imaju **konvolutivne neuronske mreže** poznate kao *CNN* (eng. *Convolutional neural network*) koje su se odlično pokazale pri radu sa slikama. Iako se mogu primeniti i na druge signale, kao što je zvuk, ograničićemo se

na rad sa slikama. Inspiracija za njihov nastanak proistekla je iz poznatog naučnog eksperimenta izvršenog 1960. godine. David Hubel i Torsten Vizel proučavali su aktivnost korteksa mozga pri posmatranju linija pod odgovarajućim uglovima. Uz pomoć mikroelektroda ugrađenih u mozak mačaka uočili su da se neuroni selektivno aktiviraju pri posmatranju linija pod odgovarajućim uglovima. Neki neuroni se najviše aktiviraju posmatrajući linije pod jednim uglom, dok se promenom ugla aktivira druga grupa neurona. Otkrili su da vizuelni deo mozga izgrađuje sliku počevši od jednostavnih elemenata ka kompleksnijim [12]. Specifična arhitektura konvolutivnih neuronskih mreža omogućava prepoznavanje sve apstraktnijih oblika izdvajajući prvo najjednostavnije. Kao i sve neuronske mreže, sastoje se od ulaznog, izlaznog i skrivenih slojeva. Specifično za *CNN* je postojanje konvolutivnih slojeva koji se zasnivaju na operaciji konvolucije.

Konvolutivni sloj

Primenom matematičke operacije konvolucije na neprekidne funkcije f i g kao rezultat se dobija nova funkcija definisana na sledeći način:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx$$

Kako su slike prestavljene vrednostima piksela posmatra se operacija konvolucije za dvodimenzione ulaze nad diskretnim poljem vrednosti. Uvedimo konvoluciju sa dvodimenzionim ulazom, a zatim i njenu diskretnu varijantu.

$$(f * g)(i,j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)g(i-x,j-y)dxdy$$
$$(f * g)(i,j) = \sum_{x=X_{min}}^{X_{max}}\sum_{y=Y_{min}}^{Y_{max}} f(x,y)g(i-x,j-y)$$

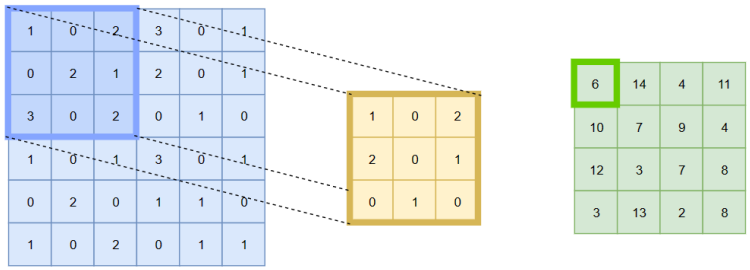
Opisane transformacije izvršavaju se u konvolutivnim slojevima neuronske mreže.

Ukoliko je na slici prikazan neki smislen objekat, postoji određena pravilnost u raspodeli piksela. Pretpostavlja se da su bliski pikseli jače korelisani od udaljenih. Primenom matematičkih operacija i kombinovanjem vrednosti okolnih piksela moguće je izdvojiti neke od pravilnosti u vidu novih slika. Kako su slike predstavljene matricom piksela, operacija konvolucije se izvršava nad dve matrice: ulaznom matricom i filterom. Filter se naziva i kernelom i on određuje obrasce koje tražimo na slici, na primer horizontalne ili vertikalne linije. U zavisnosti od filtera, primenom operacije konvolucije izvršavaju se različite transformacije ulazne matrice i izdvajaju se njena određena svojstva. U praksi je operacija konvolucije implementirana kao operacija *unakrsne korelacije*. U nastavku definišemo operaciju unakrsne korelacije:

Za ulaznu matricu X dimenzije $a \times b$ i filter F dimenzije $c \times d$ rezultat je matrica $(X * F)_{i,j}$ dimenzije $(a - c + 1) \times (b - d + 1)$ [14].

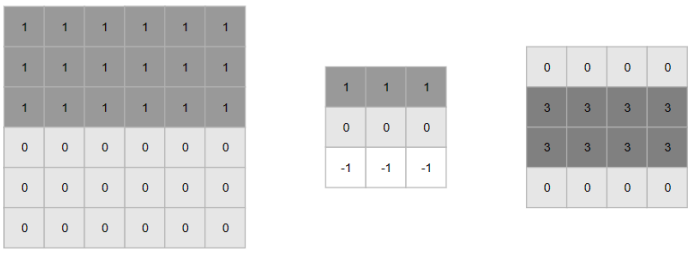
$$(X * F)_{i,j} = \sum_{k=0}^{c-1} \sum_{l=0}^{d-1} X_{i+k,j+l} F_{k,l} \qquad (i = 0, 1, \ldots, a - c \ , \ j = 0, 1, \ldots, b - d)$$

Pomeranjem matrice filtera duž ulazne matrice i računanjem skalarnog proizvoda na odgovarajućim pozicijama formira se rezultujuća matrica (Slika 2.3).

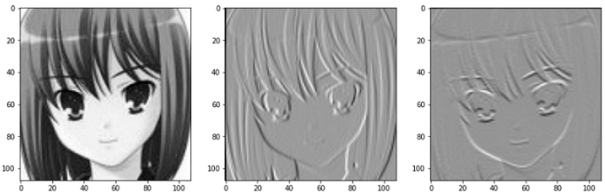


Slika 2.3: Primer operacije konvolucije. Ulazna matrica je obojena plavom bojom, filter žutom, a rezultujuća matrica zelenom bojom. Element rezultujuće matrice dobija se kao skalarni proizvod filtera i jednog dela ulazne matrice. Pomeranjem filtera duž ulazne matrice računaju se svi rezultujući elementi.

Izlazne matrice jedne slike zavise od filtera koji se koristi. Ukoliko je velika razlika između vrednosti bliskih piksela u istoj horizontali, najverovatnije se tu nalazi vertikalna ivica. Ukoliko je velika razlika između vrednosti bliskih piksela u istoj vertikali, najverovatnije se tu nalazi horizontalna ivica. Operacijom konvolucije sa odgovarajućim filterima mogu se uočiti takve razlike, samim tim i horizontalne i vertikalne ivice. Primer filtera za uočavanje horizontalnih ivica dat je matricom $\mathrm{H} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$ a vertikalnih matricom $\mathrm{V} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$ (Slika 2.5).



Slika 2.4: Primer uočavanja horizontalne linije. Filter H računa razlike piksela prve i poslednje vrste delova ulazne matrice dimenzije 3x3. Tako otkriva horizontalne linije sa čijih strana se vrste piksela jače razlikuju. Analogno filter V otkriva vertikalne.



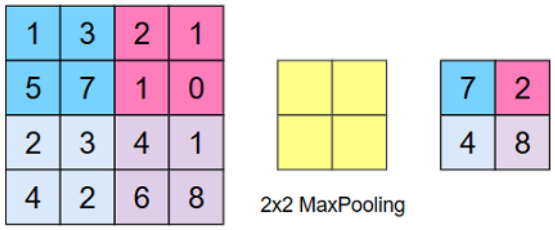
Slika 2.5: Rezultati korišćenja filtera V i H redom na sliku dimenzije 108x108 piksela

Pri obradi slika u boji prisutan je veći broj ulaznih matrica, po jedna za svaki kanal boje. Potrebno je obezbediti višekanalnu operaciju konvolucije i na svaku matricu ulazne slike primeniti filter. Tenzor je niz matrica iste dimenzije koje nazivamo kanalima. Za ulazni tenzor X dimenzije $a \times b \times f$ i filter tenzor F dimenzije $c \times d \times f$ rezultat konvolucije tenzora je matrica $(X * F)_{i,j}$ dimenzije $(a-c+1) \times (b-d+1)$ [14], dobijena na sledeći način:

$$(X * F)_{i,j} = \sum_{k=0}^{c-1} \sum_{l=0}^{d-1} \sum_{p=0}^{f-1} X_{i+k,j+l,p} F_{k,l,p} \qquad (i = 0, 1, \ldots, a-c \ , \ j = 0, 1, \ldots, b-d)$$

Sloj agregacije

Slojevi agregacije (eng. *pooling layer*) smanjuju dimenzije izlaza konvolutivnih slojeva. Jednom kanalu konvolucije odgovara jedan kanal agregacije. Svaki kanal tenzora se izdeli na delove. Potom se na izdeljene delove primenjuje funkcija agregacije, koja je najčešće neka jednostavna funkcija poput izračunavanja maksimuma ili proseka. Koliko će biti smanjen izlaz zavisi od dimenzije matrice agregacije. Ukoliko se agregiraju delovi dimenzija $n \times n$, izlaz je n^2 puta manji od ulaza. Primer agregacije izborom maksimalnog elementa prikazan je na slici 2.6.



Slika 2.6: Primer agregacije izborom maksimalnog elementa matricom agregacije dimenzije 2×2. Ulazna matrica je podeljena na delove dimenzija 2×2 i svaki deo je zamenjen svojim maksimalnim elementom. Dimenzije polazne matrice su smanjene sa 4×4 na 2×2.

Slojevi agregacije smanjuju potrebnu količinu izračunavanja, memorijsku složenost kao i broj parametara koje je potrebno naučiti. Takođe povećavaju invarijantnost modela na translacije. Ipak agregacija dovodi do gubitka informacija o lokaciji pronađenog svojstva. Ukoliko je problem takav da je samo potrebno detektovati neko svojstvo, ali ne i lokaciju, ovo ne predstavlja problem.

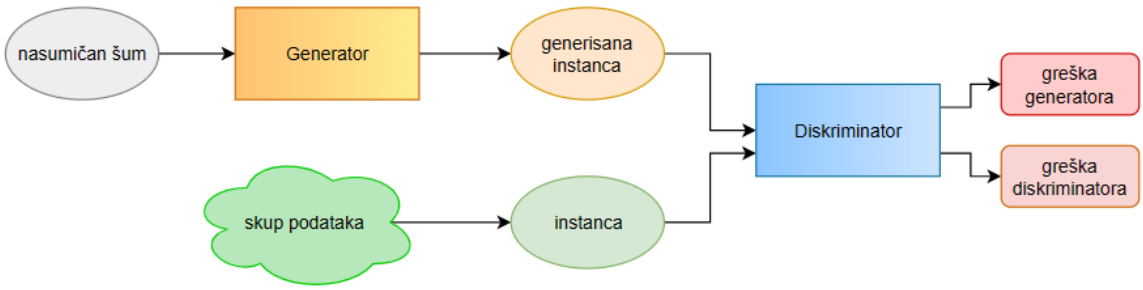
Konvolutivne neuronske mreže se sastoje od naizmeničnih slojeva konvolucije i agregacije eventualno praćenih potpuno povezanim slojem. Filteri konvolutivnih slojeva detektuju jednostavne oblike u nižim slojevima arhitekture i dolaze do složenih oblika na izlazu. Oni nisu unapred poznati i predstavljaju parametre mreže koje je potrebno naučiti. U zavisnosti od problema koji se rešava konvolutivne neuronske mreže same uče adekvatne filtere i utvrđuju koja svojstva su bitna. Konvolutivne neuronske mreže se odlikuju deljenjem parametara. Svi neuroni jednog kanala imaju iste parametre, određene filterom. Zaduženi su za računanje skalarnih proizvoda filtera na odgovarajućim pozicijama i isti parametri se uče za sve delove slike [7]. Ovo omogućava da se traženo svojstvo pronađe bilo gde na slici i smanjuje broj parametara konvolutivnih neuronskih mreža u odnosu na potpuno povezane mreže. Sve ovo doprinosi popularnosti i širokoj primeni konvolutivnih neuronskih mreža.

Glava 3

Generativne suparničke mreže

3.1 Arhitektura generativnih suparničkih mreža

Generativne suparničke mreže se sastoje iz dva dela, generatora i diskriminatora. Generator je neuronska mreža koja proizvodi nove instance iz verovatnosne raspodele trening skupa. Diskriminator je neuronska mreža koja klasifikuje podatke na generisane i podatke iz trening skupa.



Slika 3.1: GAN

Generator kao ulaz dobija nasumični šum, odnosno vektor slučajnih vrednosti. Jednostavnosti radi, u praksi se nasumični šum najčešće uzima iz uniformne ili normalne raspodele [21]. Za zadati vektor slučajnih vrednosti iz raspodele šuma p_z, generator generiše novi podatak. Izlaz generatora predstavlja ulaz za diskriminator.

Diskriminator je klasifikator čiji ulazi dolaze sa dva izvora. Ulazni skup podataka generativne suparničke mreže predstavlja stvarne podatke. Njih diskriminator posmatra kao pozitivne instance. Podaci generisani od strane generatora predstavljaju

lažne podatke, koje diskriminator posmatra kao negativne instance. Diskriminator vrši binarnu klasifikaciju i za dati ulaz daje verovatnoću da je ulaz stvaran. Na osnovu izlaza diskriminatora računaju se dve funkcije greške, greška diskriminatora i greška generatora. One redom kažnjavaju diskriminator za pogrešnu klasifikaciju podataka i generator za stvaranje podataka koji su uspešno prepoznati kao lažni.

GAN se sastoji od navedenih mreža čiji suprostavljeni ciljevi dovode do međusobnog poboljšavanja. Diskriminator razdvaja lažne instance od stvarnih. Uspešno klasifikovanje generisanih instanci kao lažnih dovodi do kažnjavanja generatora. Generator od nasumičnog šuma generiše što je moguće verodostojnije podatke, koje diskriminator neće uspeti tačno da klasifikuje.

3.2 Matematičke osnove

Osvrnimo se na matematičku osnovu generativnih suparničkih mreža. Generator i diskriminator se takmiče da postanu tačniji u svojim predviđanjima. Mreže se mogu posmatrati kao igrači koji učestvuju u igri nulte sume. U teoriji igara, nulta suma predstavlja situaciju u kojoj dobitak jednog igrača odgovara gubitku drugog. Igrači su predstavljeni diferencijabilnim funkcijama. Diskriminator je predstavljen funkcijom D koja ima parametre $\theta^{(D)}$ i ulaz x. Generator je predstavljen funkcijom G koja ima parametre $\theta^{(G)}$ i ulaz z. Svaki igrač kontrolišući svoje parametre pokušava da smanji svoju funkciju greške J. Diskriminator želi da minimizuje funkciju greške $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ kontrolišući jedino $\theta^{(D)}$. Analogno, generator minimizuje funkciju $J^{(G)}(\theta^{(G)}, \theta^{(D)})$ kontrolišući jedino $\theta^{(G)}$. Svaki igrač zavisi od parametara drugog igrača, kao što svaka mreža zavisi od napredovanja i rezultata druge mreže. Ali igrači mogu kontrolisati samo svoje parametre. Rešenje igre nulte sume je tačka *Nešovog ekvilibrijuma*. Ona predstavlja minimum funkcije greške $J^{(D)}$ u odnosu na parametre $\theta^{(D)}$, kao i minimum funkcije greške $J^{(G)}$ u odnosu na parametre $\theta^{(G)}$.

Raspodelu skupa podataka nad kojim radimo označićemo sa p_{data}, dok ćemo raspodelu generisanih podataka označiti sa p_g. Za slučajni vektor z izlaz iz generatora je $x = G(z, \theta^{(G)})$ iz raspodele p_g. Generator se obučava da na osnovu šuma z generiše instance $x = G(z, \theta^{(G)})$ za koje diskriminator $D(x, \theta^{(D)})$ neće uspeti da uspešno razlikuje raspodele $p_g(x)$ i $p_{data}(x)$.

GAN pokušava da imitira uzorke iz raspodele podataka. Zato funkcija greške generativnih suparničkih mreža mora da oslika odstojanje raspodele generisanih od raspodele stvarnih podataka. Mogu se koristiti različite funkcije grešaka.

Min-max funkcija greške je zadata na sledeći način:

$$R(D,G) = E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

Jednostavnosti radi parametri suparničkih mreža $\theta^{(G)}$ i $\theta^{(D)}$ su izostavljeni. $D(x)$ predstavlja verovatnoću sa kojom diskriminator ispravno klasifikuje stvarnu instancu x. Posmatramo očekivanje verovatnoće po svim stvarnim instancama x. $G(z)$ je izlaz generatora za slučajni vektor z. Tada $D(G(z))$ predstavlja ocenu diskriminatora da je lažna instanca $G(z)$ stvarna. Zato se u drugom sabirku posmatra očekivanje po svim nasumičnim ulazima z za $1 - D(G(z))$. Generator pokušava da smanji grešku $R(D,G)$, dok diskriminator pokušava da je poveća. Odnosno generator i diskriminator igraju igru nulte sume tokom obučavanja, vršeći $\min_G \max_D R(D,G)$. Kako prvi sabirak zavisi samo od parametara diskriminatora, generator minimizuje jedino $log(1-D(G(z)))$. Pokazano je da sa ovakom funkcijom greške GAN može da stagnira u ranim fazama dok je klasifikovanje diskriminatora lak zadatak [9]. Zato se često umesto minimizacije $log(1-D(G(z)))$ koristi maksimizacija $log(D(G(z)))$. Odnosno generator sada ne minimizuje verovatnoću ispravne klasifikacije diskriminatora, već maksimizuje verovatnoću pogrešne. Takvu funkciju greške nazivamo modifikovana min-max funkcija greške.

Formula greške se izvodi iz činjenice da je diskriminator klasifikator stvarnih i lažnih instanci. Posmatrajmo funkciju *unakrsne entropije* koja se koristi prilikom binarne klasifikacije:

$$L(u,v) = -u \log v - (1-u) \log(1-v)$$

Ulaz diskriminatora predstavljaju stvarne instance x za koje je očekivan izlaz D(x) jednak 1 i generisane instance $G(z)$ za koje se očekuje izlaz 0. Tada važi:

$$L(1, D(x)) = -\log D(x)$$
$$L(0, D(G(z))) = -\log(1 - D(G(z)))$$

Diskriminator kažnjava odstupanje $D(x)$ od 1 i odstupanje $D(G(z))$ od 0, odnosno minimizuje $-\log D(x) - \log(1 - D(G(z)))$, što je ekvivalentno maksimizovanju funkcije $R(D,G)$. Kako generator i diskriminator učestvuju u igri nulte sume, cilj generatora je da minimizuje funkciju $R(D,G)$, odnosno:

$$J^{(G)}(\theta^{(G)}, \theta^{(D)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)})$$

Igra nulte sume se često naziva min-max igra zbog forme rešenja koja sadrži minimizaciju i maksimizaciju:

$$\min_G \max_D R(D,G).$$
$$\theta^{(G)*} = arg \min_{\theta^{(G)}} \max_{\theta^{(D)}} R(\theta^{(D)}, \theta^{(G)})$$

Teorema 1. *Za fiksiran generator G optimalan diskriminator D je:*

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{3.1}$$

Dokaz. Cilj diskriminatora je maksimizacija $R(D,G)$ bez obzira na generator G

$$\begin{aligned} R(D,G) &= E_x[log(D(x))] + E_z[log(1 - D(G(z)))] \\ &= E_x[log(D(x))] + E_x[log(1 - D(x))] \\ &= \int_x (p_{data}(x)log(D(x)) + p_g(x)log(1 - D(x)))dx \end{aligned}$$

Funkcija $y \to alog(y) + blog(1-y)$ za svaki $(a,b) \in \mathbb{R}^2 \backslash (0,0)$ dostiže maksimum na intervalu $[0,1]$ u tački $\frac{a}{a+b}$. Odavde za $a = p_{data}(x)$ i $b = p_g(x)$ sledi dokaz tvrđenja. □

Za optimalan generator G generisana raspodela p_g je približna raspodeli p_{data}.

$$p_g = p_{data} \Rightarrow D^*_G(x) = \frac{1}{2} \tag{3.2}$$

Tada diskriminator nije u stanju da odluči da li je instanca x stvarna ili lažna, već vrši nasumično predviđanje.

Postavlja se pitanje kada je generisana raspodela p_g približna raspodeli p_{data}. Definišimo funkcije za ocenjivanje različitosti dve raspodele podataka P i Q. Funkcija KL divergencije (eng. *Kullback–Leibler divergence*) meri odstupanje jedne raspodele podataka od druge. Za date raspodele P i Q definisana je na sledeći način:

$$KL(P \parallel Q) = \int_x P(x) log \frac{P(x)}{Q(x)} dx$$

Ukoliko su P i Q diskretne raspodele nad istim prostorom elementarnih ishoda X formula je zadata na sledeći način:

$$KL(P \parallel Q) = \sum_{x \in X} P(x) log \frac{P(x)}{Q(x)} dx$$

Intuitivno, ukoliko je verovatnoća instance x u raspodeli P velika, a u raspodeli Q mala ili obrnuto, tada je odstupanje veliko. Odstupanje je najmanje kada je $P = Q$ i tada funkcija ima vrednost 0 i dostiže minimum. Data funkcija nije simetrična, odnosno važi:

$$KL(P \parallel Q) \neq KL(Q \parallel P)$$

Funkcija JS divergencije (eng. *Jensen-Shannon divergence*) je simetrična, nenegativna funkcija za određivanje različitosti dve raspodele podataka. Za date raspodele P i Q definisana je na sledeći način:

$$JS(P \parallel Q) = \tfrac{1}{2}KL(P \parallel \tfrac{P+Q}{2}) + \tfrac{1}{2}KL(Q \parallel \tfrac{P+Q}{2})$$

Teorema 2. *Globalni minimum funkcije* $\max_D R(D,G)$ *postiže se ako i samo ako je* $p_{data} = p_g$ *i tada funkcija ima vrednost -* $log4$.

Dokaz. Maksimum se dostiže za optimalan generator D^*_G.

$$\begin{aligned}
\max_D R(D,G) &= R(D, D^*_G) \\
&= E_{x\sim p_{data}}[log(D^*_G(x))] + E_{x\sim p_g}[log(1 - D^*_G(x))] \qquad (3.1)\\
&= E_{x\sim p_{data}}[log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)})] + E_{x\sim p_g}[log(1 - \frac{p_{data}(x)}{p_{data}(x) + p_g(x)})] \\
&= \int_x p_{data}(x) log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)})dx + \int_x p_g(x) log(1 - \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) \\
&= \int_x p_{data}(x) log(\frac{p_{data}(x)}{2(\frac{p_{data}(x)+p_g(x)}{2})})dx + \int_x p_g(x) log(\frac{p_g(x)}{2(\frac{p_{data}(x)+p_g(x)}{2})}) \\
&= KL(p_{data} \parallel \frac{p_{data} + p_g}{2}) + KL(p_g \parallel \frac{p_{data} + p_g}{2}) - 2log2 \\
&= JS(p_{data} \parallel p_g) - log4
\end{aligned}$$

Na osnovu (3.2) za $p_{data} = p_g \Rightarrow D^*_G = \frac{1}{2}$. Tada je $R(D, D^*_G) = -log4$.
Funkcija JS divergencije je nenegativna funkcija čiji minimum se dostiže ukoliko su raspodele podataka jednake. Minimum za $JS(p_{data} \parallel p_g)$ se dostiže ukoliko je $p_{data} = p_g$ i iznosi 0. Na osnovu ovoga globalni minimum funkcije $\max_D R(D,G)$ se dostiže za $p_{data} = p_g$ i iznosi $-log4$.

□

3.3 Proces obučavanja GAN-a

Kao što je već objašnjeno, funkcije D i G učestvuju u sledećoj min-max igri:

$$\min_G \max_D E_x[log(D(x))] + E_z[log(1 - D(G(z)))].$$

Igra se sastoji od dva naizmenična koraka:
Korak 1: Za fiksirano G izvodi se gradijentni spust za maksimizaciju:

$$\max_D E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

Korak 2: Za fiksirano D izvodi se gradijentni spust za minimizaciju:

$$\min_G E_x[log(D(x))] + E_z[log(1 - D(G(z)))] \iff \min_G E_z[log(1 - D(G(z)))]$$

U modifikovanoj min-max igri vrši se $\max_G E_z[log(D(G(z)))]$.

U nastavku je prikazan algoritam obučavanja generativnih suparničkih mreža stohastičkim gradijentnim spustom koji je predložio Ian Gudfelov [9].

Algoritam 1 Algoritam obučavanja generativnih suparničkih mreža

Napomena: broj koraka koje primenjujemo na diskriminatoru je hiperparametar k

for broj iteracija **do**

for k koraka **do**

- Uzorkovati podskup od m elemenata $\{z_1, ..., z_m\}$ iz raspodele $p_g(z)$
- Uzorkovati podskup od m elemenata $\{x_1, ..., x_m\}$ iz raspodele $p_{data}(x)$
- Ažurirati parametre diskriminatora na osnovu stohastičkog gradijenta

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + log(1 - D(G(z_i)))]$$

end for

- Uzorkovati podskup od m elemenata $\{z_1, ..., z_m\}$ iz raspodele $p_g(z)$
- Ažurirati parametre generatora na osnovu stohastičkog gradijenta

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [log(1 - D(G(z_i)))]$$

end for

U praksi generator i diskriminator su predstavljeni neuronskim mrežama.

Obučavanje diskriminatora: Diskriminator klasifikuje stvarne i generisane podatke. Na osnovu klasifikacije izračunavaju se dve funkcije greške: greška diskriminatora i greška generatora. Prilikom obučavanja diskriminatora, greška generatora se zanemaruje. Na osnovu greške diskriminatora algoritmom propagacije unazad računaju se gradijenti i menjaju se težine parametara diskriminatora.

Obučavanje generatora: Ulaz predstavlja nasumični šum na osnovu kog generator generiše instance (u našem slučaju slike). Generator nije direktno povezan sa svojom funkcijom greške, već dobija povratnu informaciju od diskriminatora. Izlaz iz generatora prosleđuje se diskriminatoru čiji trening skup pored generisanih slika sadrži i stvarne. Na osnovu rezultata klasifikacije računaju se opisane greške. Greška generatora se algoritmom propagacije unazad kroz diskriminator prenosi do generatora i računaju se gradijenti. Izračunati gradijenti se koriste za izmenu težina parametara generatora. Vrši se kažnjavanje generatora za generisanje instanci koje je diskriminator uspešno prepoznao kao lažne.
Proces obučavanja generativnih suparničkih mreža sastoji se od naizmeničnog treniranja dve suprostavljene mreže. Zato sledeći postupak obučavanja teško konvergira:

1. Diskriminator trenira jednu ili više uzastopnih epoha. Generator je fiksiran prilikom treniranja diskriminatora.

2. Generator trenira jednu ili više uzastopnih epoha. Diskriminator se ne menja tokom treniranja generatora.

3. Ponavljaju se koraci 1. i 2.

Na početku diskriminator lako prepoznaje stvarne instance od generisanih. Tokom treniranja generator postaje sve bolji u generisanju lažnih instanci. Ukoliko je impresivna verodostojnost izgenerisanih slika, tačnost predviđanja diskriminatora je 50%, odnosno diskriminator baca novčić prilikom klasifikacije instanci.

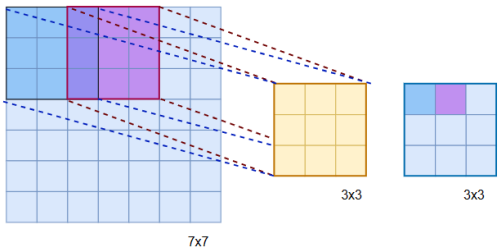
3.4 DCGAN

Prilikom obučavanja generativnih suparničkih mreža za generisanje slika bolji rezultati su očekivani ukoliko su neuronske mreže specijalizovane za rad nad slikama. Duboke konvolutivne generativne suparničke mreže, poznatije pod imenom DCGAN (eng. *Deep Convolutional GAN*) predstavljaju podvrstu generativnih suparničkih mreža u kojoj su generator i diskriminator konvolutivne neuronske mreže.

Mnogi pokušaji da se konvolutivne neuronske mreže uklope u strukturu generativnih suparničkih mreža završile su neuspehom. Predložene su sledeće izmene u arhitekturi konvolutivnih mreža kako bi se postiglo što stabilnije obučavanje *GAN*-a:

1. Zamena slojeva agregacije slojevima konvolucije sa zadatom veličinom koraka
2. Korišćenje unutrašnje standardizacije [13] u generatoru i diskriminatoru
3. Korišćenje *ReLU* aktivacione funkcije u svim slojevima generatora, osim u poslednjem u kom se koristi *Tanh*.
4. Korišćenje *LeakyReLU* aktivacione funkcije u slojevima diskriminatora [18]

U drugom poglavlju je objašnjeno kako slojevi agregacije mogu smanjiti dimenzije izlaznih matrica. Sličan efekat se postiže korišćenjem konvolutivnih slojeva uz zadatu veličinu koraka (eng. *stride*). Korak je broj piksela za koji se filter pomera u horizontalnom ili vertikalnom smeru prilikom konvolucije. Podrazumevani korak pomeraja filtera preko ulazne matrice je veličine jedan u oba smera, odnosno (1,1). Izmena koraka dovodi do izmene veličine izlazne matrice. Ukoliko je korak izmenjen na $(2, 2)$ filter se pri svakom vertikalnom pomeraju pomera za dve pozicije nadole, a pri svakom horizontalnom po dve pozicije udesno. Na slici 3.2 operacijom konvolucije filterom dimenzije 3×3 sa pomerajem $(2, 2)$ dimenzija matrice je smanjena sa 7×7 na 3×3.



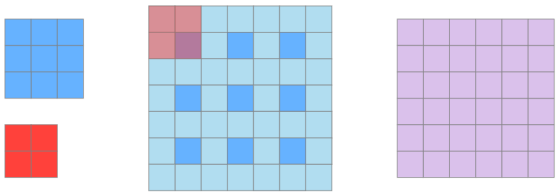
Slika 3.2: Primer konvolucije sa pomerajem $(2, 2)$

Ukoliko dimenzija ulazne matrice nije deljiva veličinom filtera vrši se proširivanje ulazne matrice nulama (eng. *padding*). Označimo veličinu proširivanja nulama sa P, veličinu koraka udesno sa S_d i koraka nadole sa S_n, visinu i širinu ulazne matrice sa H i W redom, visinu i širinu filtera sa H_f i W_f redom. Širina W_o i visina H_o izlazne matrice se računa prema sledećim formulama:

$$H_o = \frac{H - H_f + 2P}{S_n} + 1 \ , \ W_o = \frac{W - W_f + 2P}{S_d} + 1$$

Prilikom klasifikacije, na opisan način se slojevima konvolucije vrši smanjivanje dimenzija. Sa druge strane zadatak generatora je da od nasumičnih vrednosti generiše slike, odnosno matrice odgovarajućih dimenzija. Da bi ovo bilo moguće mora

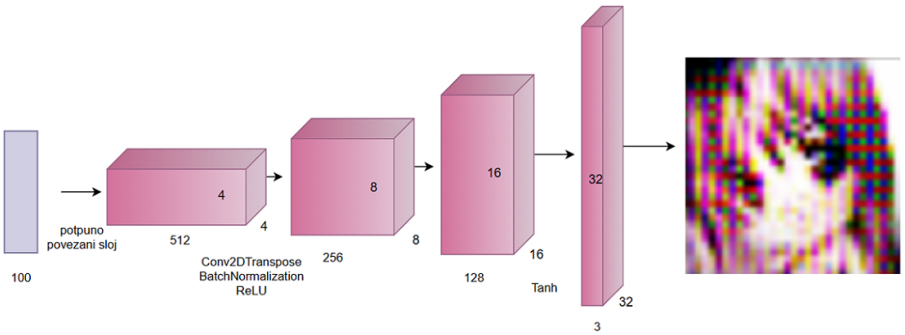
postojati korak suportan sažimanju koji će uvećavati dimenzije izlaznih matrica. Željeni rezultati se dobijaju operacijom **transponovane konvolucije** (Slika 3.3).



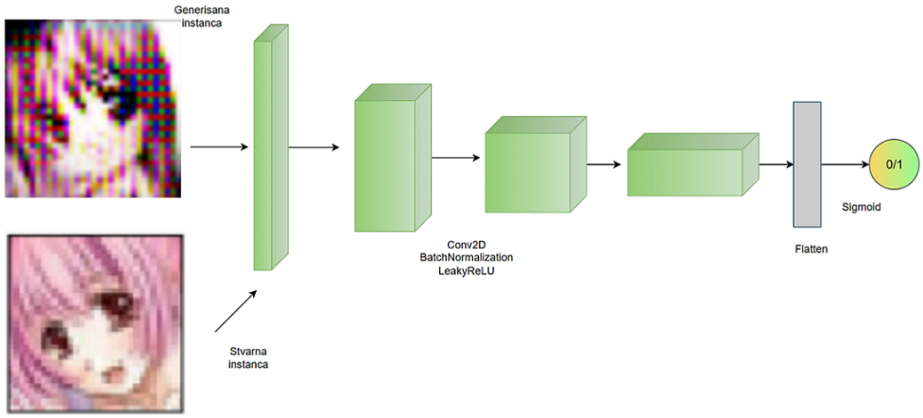
Slika 3.3: Primer transponovane konvolucije

Posmatrajmo primer sa slike 3.3. Operacija je primenjena na ulaz dimenzije 3×3 sa filterom dimenzije 2×2 i korakom veličine 2. Transponovana konvolucija zahteva dodatni korak u odnosu na operaciju konvolucije. Prvo se na osnovu elemenata ulazne matrice kreira matrica međurezultata. Za razliku od operacije konvolucije, prilikom transponovane konvolucije veličina koraka određuje koliko su udaljeni elementi ulazne matrice u matrici međurezultata i ne predstavlja veličinu pomeraja filtera. Elementi ulazne matrice su razdvojeni nulama. Ostatak postupka je isti kao kod primene operacije konvolucije na matricu međurezultata. Na osnovu ulaza dimenzije 3×3 dobija se izlaz dimenzije 6×6.

Zamenom slojeva agregacije konvolutivnim slojevima generativna suparnička mreža sama uči odgovarajuće funkcije agregacije [17]. Na slikama 3.4 i 3.5 prikazani su primeri strukture generatora i diskriminatora DCGAN modela.



Slika 3.4: Primer DCGAN generatora. Nakon potpuno povezanog sloja, nizom slojeva transponovane konvolucije menja se dimenzija ulaznog tenzora. Koristi se unutrašnja standardizacija i *ReLU* aktivaciona funkcija u svim slojevima osim u poslednjem, koji koristi *Tanh* aktivacionu funkciju. Na osnovu vektora nasumičnih vrednosti veličine 100 generiše se slika dimenzije 32x32x3.



Slika 3.5: Primer DCGAN diskriminatora. Ulaz predstavljaju generisane i stvarne slike dimenzija 32x32x3. Slojevima konvolucije menja se dimenzija tenzora. Koristi se unutrašnja standardizacija i $LekyReLU$ aktivaciona funkcija. U poslednjem sloju, nakon prevođenja višedimenzionalnog tenzora u jednodimenzioni primenjuje se sigmoidna aktivaciona funkcija. Izlaz je vrednost iz intervala $[0, 1]$ koja određuje pripadnost odgovarajućoj klasi.

Zbog vizuelno zadovoljavajućih rezultata prilikom rešavanja problema generisanja slika, DCGAN predstavlja osnovu za najpopularnije vste generativnih modela. Pokazao se kao dobar model za generisanje slika visoke rezolucije bez potrebe za uvođenjem velikog broja konvolutivnih slojeva.

Glava 4

Izazovi obučavanja GAN-ova

4.1 Problemi i predložena rešenja

Proces obučavanja generativnih suparničkih mreža predstavlja težak problem za koji još uvek nije pronađen algoritam koji garantuje sigurnu konvergenciju. Kako generator i diskriminator učestvuju u igri nulte sume, gubitak jednog igrača predstavlja dobitak drugog i obrnuto. Obučavanje generativnih suparničkih mreža ne garantuje dostizanje Nešovog ekvilibrijuma u opisanoj igri. Samim tim zadatak generisanja slika uz pomoć GAN modela neretko rezultuje vizuelno nezadovoljavajućim generisanim slikama.

Problem konvergencije

Posmatrajmo algoritam treniranja suparničkih mreža gradijentnim spustom na jednostavnom primeru. Funkcija greške zadata je sa $L(x, y) = xy$. U igri nulte sume generator pokušava da minimizuje funkciju $l_1(x) = xy$, dok diskriminator pokušava da minimizuje funkciju $l_2(y) = -xy$. Ažuriranje parametara vrši se gradijentnim spustom i oba igrača menjaju jedino svoje parametre. Kako je

$$\frac{\partial l_1}{\partial x} = y \ , \ \frac{\partial l_2}{\partial x} = -x$$

za brzinu učenja α igrači vrše sledeća ažuriranja parametara:

$$x \leftarrow x - \alpha y \ , \ y \leftarrow y + \alpha x$$

Svaki igrač smanjuje odgovarajuću funkciju greške i tako implicitno utiče na funkciju greške drugog igrača. Navedeni postupak dovodi do stabilnih kružnih oscilacija parametara i ne konvergira ka željenom ekvilibrijumu $x = y = 0$ [19]. Na navedenom

primeru uočava se problem koji se često javlja prilikom obučavanja GAN-ova. Generator i diskriminator mogu zauvek uvećavati i smanjivati funkciju greške $R(D, G)$ ne dostižući tačku Nešovog ekvilibrijuma. Opisani problem predstavlja problem konvergencije generativnih suparničkih mreža i javlja se prilikom istovremenog treniranja generatora i diskriminatora metodom gradijentnog spusta.

Kako generator napreduje povratna informacija od diskriminatora postaje sve manje značajna. U situaciji kada diskriminator ne ume da razlikuje stvarne od generisanih podataka, tačnost diskriminatora iznosi 50% i vrši se nasumično klasifikovanje. Nastavak obučavanja generatora nad beskorisnom povratnom informacijom diskriminatora dovodi do ponovnog pada njegovog kvaliteta. U praksi se teško postiže stabilna konvergencija i balans između treniranja generatora i diskriminatora.

Problem nestajućih gradijenata

Problem nestajućih gradijenata generatora javlja se kada je diskriminator veoma tačan prilikom klasifikacije podataka. Algoritmom propagacije unazad računamo gradijente krećući se unazad kroz slojeve, od poslednjeg do prvog. Množenjem izvoda prilikom prolaska kroz slojeve vrednost gradijenta postaje sve manja. Optimalan diskriminator ne daje dovoljno povratnih informacija na osnovu kojih generator može da uči. Veoma male vrednosti izračunatih gradijenata onemogućavaju dalje napredovanje generatora. Kada bi diskriminator D uvek tačno klasifikovao podatke, odnosno ukoliko bi važilo $D(x) = 1$, $\forall x \in p_{data}$ i $D(G(z)) = 0$, $\forall z \in p_z$, funkcija greške $R(D, G) = E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$ i izračunati gradijenti približili bi se nuli i zaustavili obučavanje generatora [24].

U ranim fazama obučavanja, kada je razlika između stvarnih i generisanih slika jasno uočljiva, često se javlja problem nestajućih gradijenata. Modifikovana minmax funkcija greške obezbeđuje veće vrednosti gradijenta pri početku treniranja [9]. Kao što već znamo generator tada maksimizuje funkciju $log(D(G(z)))$ umesto minimizacije funkcije $log(1 - D(G(z)))$. Odnosno umesto prvobitne funkcije greške generatora zadate formulom:

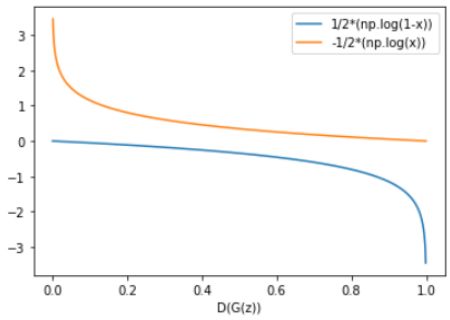
$$J^{(G)} = \frac{1}{2} E_z log(1 - D(G(z)))$$

koristi se nova funkcija greške:

$$J^{(G)} = -\frac{1}{2} E_z log(D(G(z)))$$

Kada je $D(G(z))$ približno 0 klasifikacija je uspešna i pokušaj generatora da prevari diskriminator nije uspeo. Na grafiku sa slike 4.1 vidi se da u tim situacijama imamo

veoma male vrednosti gradijenata za prvobitnu funkciju greške, kako smo smanjivali verovatnoću ispravne klasifikacije. Nova funkcija greške uvećava verovatnoću pogrešne klasifikacije i obezbeđuje veće vrednosti gradijenata u ranim fazama treniranja. Iako se često primenjuje, modifikovana funkcija greške ne garantuje uspešnost obučavanja.



Slika 4.1: Funkcije greške generatora. Modifikovana funkcija greške obezbeđuje veće vrednosti gradijenta kada je vrednost $D(G(z))$ blizu nule.

Kolaps modela

Algoritam gradijentnog spusta nije sposoban da utvrdi da li je pronađeni minimum lokalni ili globalni. Ukoliko se generator i diskriminator nađu u okruženju u kom ne mogu da smanje svoje gubitke, postoji mogućnost pronalaženja lokalnog Nešovog ekvilibrijuma [11].

Za različite ulaze očekuje se da generativna suparnička mreža proizvodi različite izlaze. Poželjno je da generisani podaci obuhvate celokupnu raspodelu podataka, uključujući i retke scenarije. Kolaps modela (eng. *mode collapse*) predstavlja situaciju u kojoj generator uvek proizvodi jednan isti podatak ili mali podskup podataka. Generator može da nauči da su određene instance prihvatljivije od drugih, odnosno da ih pogrešno klasifikuje kao stvarne. Tada se generator preprilagođava i počinje da generiše stalno iste izlaze, kako bi prevario diskriminator. Ovo onemogućava dalje napredovanje diskriminatora i model ostaje zaglavljen u lokalnom minimumu. Prilikom delimičnog kolapsa modela, generisani podaci su raznovrsniji ali sadrže zajedničke odlike i ne obuhvataju celokupnu raznovrsnost polaznog skupa podataka.

Jedan od prvobitnih razloga nastanka kolapsa modela bilo je istovremeno obučavanje generatora i diskriminatora gradijentnim spustom. Rešenje min-max igre različito je od rešenja max-min igre. Optimalan model generatora u igri

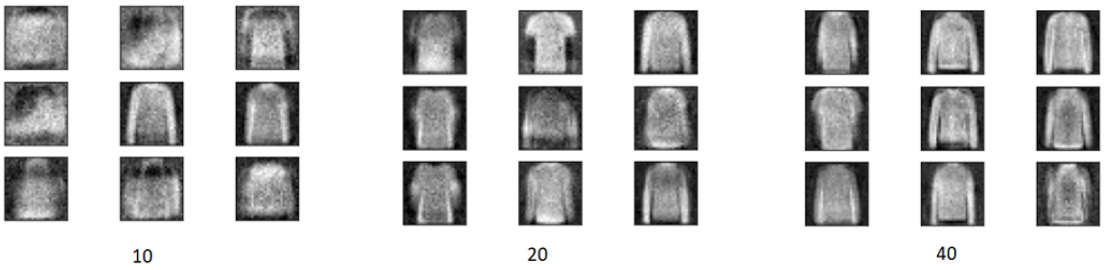
$$G^* = \min_G \max_D R(G, D)$$

generiše podatke iz željene raspodele. Promenom redosleda traženja maksimuma i minimuma, traženje minimuma se vrši u unutrašnjoj petlji i dobijeni generator

$$G^* = \max_D \min_G R(G, D)$$

svaku vrednost z preslikava u instancu x koju diskriminator sa velikom verovatnoćom pogrešno klasifikuje kao lažnu. Istovremeno obučavanje modela gradijentnim spustom ne garantuje da će se primenjivati min-max umesto max-min pristupa [8].

Glavni uzrok kolapsa modela predstavlja preprilagođavanje generatora u odnosu na fiksirani diskriminator. Generisanje instanci, koje diskriminator sa velikom sigurnošću klasifikuje kao stvarne, dovodi do preprilagođavanja oba modela i kolapsa celokupne strukture. Primer kolapsa modela nad *Fashion-MNIST* skupom podataka prikazan je na slici 4.2. Skup podataka nad kojim je vršeno obučavanje sastoji se od 60000 slika odevnih predmeta koje pripadaju jednoj od 10 mogućih klasa.



Slika 4.2: Primer kolapsa modela prilikom obučavanja GAN-a. Posmatran je uzorak generisanih slika nakon redom 10, 20 i 40 epoha treniranja. U ranim fazama obučavanja primetna je veća raznovrsnost i slabiji kvalitet generisanih slika. Nakon 40 epoha uzorkovane generisane slike pokrivaju dve od mogućih deset klasa.

Ukoliko se generator predugo trenira bez ažuriranja parametara diskriminatora nastaje opisani problem kolapsa modela. Sa druge strane ukoliko se diskriminator predugo trenira bez napredovanja generatora, može nastati problem nestajućih gradijenata. Ne postoji garancija da će proces obučavanja konvergirati niti da će uspešno biti pronađena tačka Nešovog ekvilibrijuma. Naizmeničnim obučavanjem generatora i diskriminatora pokušavamo da izbegnemo navedene probleme. Ipak određivanje ravnoteže između obučavanja diskriminatora i obučavanja generatora je težak zadatak i često završava neuspehom. U nastavku su navedene neke od predloženih tehnika za rešavanje opisanih problema.

Tehnike za postizanje stabilnog obučavanja

1. Dodavanje šuma

Kako bi se sprečila preprilagođenost diskriminatora dodaje se nasumičan šum na njegove ulaze. Ukoliko je diskriminator siguran u svojim predviđanjima povratna informacija nije korisna i javljaju se nestajući gradijenati. Dodavanjem šuma na stvarne podatke otežava se zadatak diskriminatora.

2. Jednostrano skaliranje ciljnih promenljivih

Jednostrano skaliranje vrednosti ciljnih promenljivih (eng. *one-sided label smoothing*) predstavlja regularizacionu tehniku koja sprečava preprilagođavanje diskriminatora. Vrednosti pozitivnih instanci umanjuju se za mali pozitivan faktor α. Odnosno umesto vrednosti 1 za pozitivne instance diskriminator dobija vrednost $1-\alpha$. Uobičajno se koristi vrednost 0.1 za faktor α. Intuitivno smanjuje se sigurnost sa kojom diskriminator tvrdi da je neka instanca stvarna. Skaliranje je jednostrano i vrednosti negativnih instanci ostaju nepromenjene. Pretpostavimo da je ciljna vrednost $1-\alpha$ za stvarne podatke i $0+\beta$ za generisane. Tada je optimalan diskriminator zadat sa:

$$D^*(x) = \frac{(1-\alpha)p_{data}(x)+\beta p_g(x)}{p_{data}(x)+p_g(x)}$$

Kada je faktor β jednak 0, rezultat procesa skaliranja je smanjenje optimalne vrednosti diskriminatora. Za nenula vrednosti faktora β funkcija optimalnog diskriminatora menja svoj oblik [8]. Optimalan diskriminator uspešno prepoznaje instancu kao lažnu kada je $p_{data}(x)$ blizu nule i $p_g(x)$ ima veliku vrednost. Prisustvo $p_g(x)$ u deljeniku funkcije optimalnog diskriminatora predstavlja problem u ovakvim situacijama. Zato je važno da skaliranje bude izvršeno jedino nad stvarnim instancama.

3. Unutrašnja standaradizacija

Standardizacija ulaza tako da prosek bude centriran oko nule i sa varijansom jedan ima presudnu ulogu u postizanju stabilnijeg i bržeg obučavanja neuronskih mreža. Kod dubokih neuronskih mreža izlazi jednog sloja predstavljaju ulaze drugog, pa ih je poželjno standardizovati. Zato se u slučaju DCGAN modela dodaju slojevi unutrašnje standardizacije (eng. *batch normalization*). Na nivou podskupa na kom se računaju gradijenti vrši se standardizacija izlaza svake jedinice na sledeći način:

$$y_i = \frac{x_i-\mu_B}{\sigma_B}\gamma + \beta$$

gde μ_B i σ_B predstavljaju prosek i standardnu devijaciju izlaza. Opcioni faktori skaliranja i translacije γ i β, sa podrazumevanim vrednostima 1 i 0, predstavljaju parametre koje je moguće naučiti. Kao rezultat izlaz ima prosek β i standardnu devijaciju γ [25]. Iako smanjuje verovatnoću nastanka kolapsa modela, primena unutrašnje standardizacije na sve slojeve dovodi do nestabilnosti modela. Kako bi se ovo izbeglo unutrašnja standardizacija se ne primenjuje na izlazni sloj generatora i ulazni sloj diskriminatora [18].

4. Nova funkcija greške

Prilikom kolapsa modela i problema nestajućih gradijenata predlaže se korišćenje drugih funkcija greške, kao što je Vaserštajnova (eng. *Wasserstein*) funkcija greške.

4.2 WGAN i WGAN-GP

Kako želimo da što je više moguće približimo generisanu raspodelu p_g raspodeli podataka p_{data}, potrebne su nam metrike koje će oceniti udaljenost između dve raspodele. Do sada smo se upoznali sa funkcijama KL i JS divergencije. Ispostavlja se da nedostatak neprekidnosti ovih divergencija može dovesti do poteškoća prilikom obučavanja. Zato ćemo uvesti novu metriku zvanu Vaserštajnova udaljenost.

Vaserštajnova udaljenost poznatija pod nazivom *„earth-mover"* (EM) udaljenost, zadata je sledećom formulom:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma}[\|x - y\|]$$

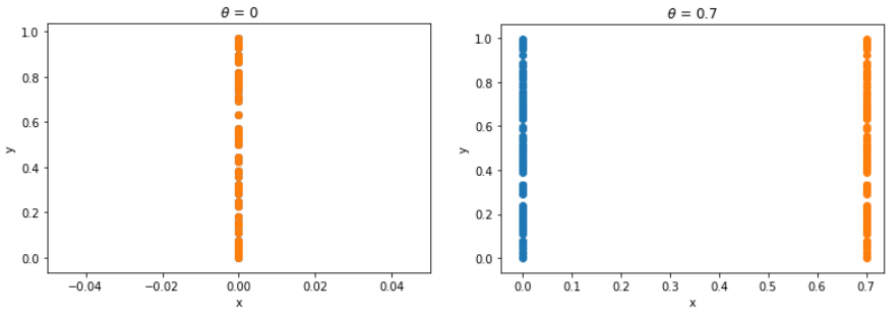
gde su P_r i P_g raspodele čije odstupanje tražimo, a $\Pi(P_r, P_g)$ je skup svih zajedničkih raspodela γ čije su marginalne raspodele P_r i P_g [2].

Ukoliko raspodele P_r i P_g zamislimo kao dve hrpe zemlje, Vaserštajnova udaljenost predstavlja minimalan trošak potreban za transformisanje jedne hrpe zemlje u drugu. Trošak pomeranja mase m za udaljenost s iznosi $m \cdot s$. Izračunavanje EM udaljenosti je optimizacioni problem pri kojem je potrebno pronaći optimalan plan transporta zemlje. Intuitivno, svako $\gamma \in \Pi(P_r, P_g)$ predstavlja jedan plan transporta. U tom slučaju $\gamma(x, y)$ određuje količinu zemlje premeštenu sa tačke x u tačku y za plan transporta γ. Kako je udaljenost između tačaka x i y jednaka $\|x - y\|$, trošak jedne ovakve transformacije iznosi $\gamma(x, y) \cdot \|x - y\|$. Kako se svaki transportni plan sastoji od niza ovakvih transformacija, potrebno je pronaći onaj sa ukupnim

minimalnim troškom po svim parovima (x, y) [23]. To jest u slučaju neprekidnih raspodela P_r i P_g potrebno je pronaći:

$$\inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma}[\|x - y\|] = \inf_{\gamma \in \Pi(P_r, P_g)} \int_x \int_y \gamma(x, y)\|x - y\| dy dx$$

Primer 4.2.1. Neka je P_0 raspodela slučajne promenljive $(0, Z) \in \mathbb{R}^2$ i P_θ raspodela slučajne promenljive $(\theta, Z) \in \mathbb{R}^2$ za fiksiran realni parametar θ i $Z \sim U(0, 1)$.



Slika 4.3: Raspodele P_0 i P_θ za $\theta = 0$ i $\theta = 0.7$

Sa slike 4.3 se vidi da se za vrednost parametra $\theta = 0$ raspodele P_0 i P_θ u potpunosti preklapaju. Za $\theta \neq 0$ ne postoji preklapanje i tada važi:

$$KL(P_0 \parallel P_\theta) = \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \tfrac{1}{0} = +\infty$$

Jednostavnim računom dobijaju se naredna tvrđenja:

$$KL(P_0 \parallel P_\theta) = KL(P_\theta \parallel P_0) = \begin{cases} +\infty, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases} \tag{4.1}$$

$$JS(P_0 \parallel P_\theta) = \begin{cases} \log 2, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases} \tag{4.2}$$

$$W(P_0, P_\theta) = |\theta| \tag{4.3}$$

Iz navedenog primera se vidi da uvođenje Vaserštajnove udaljenosti može dovesti do stabilnijeg obučavanja gradijentnim spustom. Do sada korišćena JS divergencija ima nagli skok kada $\theta \to 0$ i u tački $\theta = 0$ nije ni neprekidna ni diferencijabilna. Za Vaserštajnovu metriku niz raspodela $(P_{\theta_t})_{t\in\mathbb{N}}$ konvergira ka P_0 kada $\theta_t \to 0$, dok u slučaju JS i KL divergencije ne konvergira uopšte [2].

Planova transporta $\gamma \in \Pi(P_r, P_g)$ može biti potencijalno beskonačno mnogo. Ispitivanje svih njih zarad pronalaženja infimuma je najčešće nemoguće. Iz *Kantorovich-Rubinstein* dualnosti izvodi se drugi oblik formule Vaserštajnove udaljenosti:

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} E_{x \sim P_r}[f(x)] - E_{x \sim P_g}[f(x)] \tag{4.4}$$

gde se supremum posmatra po svim 1-Lipšic neprekidnim funkcijama $f : X \to \mathbb{R}$. Za funkciju $f : X \to Y$ kažemo da je K-Lipšic neprekidna ukoliko važi:

$$d_y(f(x_1), f(x_2)) \leq K d_x(x_1, x_2) \ , \forall x_1, x_2 \in X$$

gde su d_y i d_x redom metrike nad metričkim prostorima X i Y [1].
Pojednostavljeni oblik uslova za 1-Lipšic neprekidnost funkcije glasi:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2| \ , \forall x_1, x_2 \in X$$

Kao što je na početku objašnjeno, Vaserštajnovom metrikom želimo da ocenimo različitost raspodele podataka p_{data} i generisane raspodele p_g.

$$\begin{aligned} W(p_{data}, p_g) &= \inf_{\gamma \in \Pi(p_{data}, p_g)} E_{(x,y) \sim \gamma}[\|x - y\|] \\ &= \inf_{\gamma \in \Pi(p_{data}, p_g)} E_{(x, G(z)) \sim \gamma}[\|x - G(z)\|] \\ &= \sup_{\|f\|_L \leq 1} E_{x \sim p_{data}}[f(x)] - E_{z \sim p_z}[f(G(z))] \\ &= \max_{w \in \mathcal{W}} E_{x \sim p_{data}}[f_w(x)] - E_{z \sim p_z}[f_w(G(z))] \end{aligned} \tag{4.4}$$

U navedenoj formuli supremum se aproksimira maksimumom po parametrizovanoj familiji 1-Lipšic neprekidnih funkcija $\{f_w\}_{w \in \mathcal{W}}$. Intuitivno, f_w je jedna od mogućih funkcija diskriminatora.

WGAN (*Wasserstein GAN*) je vrsta GAN modela, kod koje je min-max igra zadata na sledeći način:

$$\min_G \max_D E_{x \sim p_{data}}[D(x)] + E_{z \sim p_z}[D(G(z))]$$

Bolji naziv za diskriminator je sada kritičar. Cilj diskriminatora nije više da uspešno klasifikuje podatke na stvarne i generisane, već da vrati procenu koliko verodostojno izgledaju prosleđene slike.
Funkcija greške diskriminatora zadata je sledećom formulom:

$$L_{critic}(w) = \max_{w \in \mathcal{W}} E_{x \sim p_{data}}[f_w(x)] - E_{z \sim p_z}[f_w(g_\theta(z))] \tag{4.5}$$

Kako bi se postiglo svojstvo K-Lipšic neprekidnosti funkcija $\{f_w\}_{w\in\mathcal{W}}$, izvršava se operacija odsecanja težina w (eng. *weight clipping*). Nakon svakog ažuriranja vrši se ograničavanje vrednosti težina unutar intervala $[-c, c]$, gde c predstavlja faktor odsecanja. Odsecanje težina ne predstavlja najbolji način obezbeđivanja Lipšic neprekidnosti, ali se koristi zbog svoje jednostavnosti i dobrih performansi. Mali faktor odsecanja može dovesti do problema nestajućih gradijenata u slučaju kada je prisutan veliki broj slojeva ili kada nije korišćena unutrašnja standardizacija. Veliki faktor odsecanja usporava proces treniranja diskriminatora do optimalnog [2].

Proces obučavanja WGAN-ova predložen u originalnom radu zadat je narednim algoritmom. Korišćene podrazumevane vrednosti hiperparametara: $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{critic} = 5$ [2].

Algoritam 2 Algoritam obučavanja WGAN-ova

Hiperparametri: α-brzina učenja algoritma, c-faktor odsecanja, m-veličina podskupa podataka, n_{critic}-broj iteracija diskriminatora po jednoj iteraciji generatora
Parametri: w-parametri diskriminatora(w_0 inicijalna vrednost), θ-parametri diskriminatora(θ_0 inicijalna vrednost)
Napomena: g_θ i g_w-gradijenti za θ i w, $RMSProp$-optimizacijski algoritam

while θ ne konvergira **do**
  for $t = 0, ..., n_{critic}$ **do**
    - Uzorkovati podskup od m elemenata $\{z_1, ..., z_m\}$ iz raspodele p_z
    - Uzorkovati podskup od m elemenata $\{x_1, ..., x_m\}$ iz raspodele p_{data}
    - Ažurirati parametre diskriminatora sledećim postupkom:

      $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
      $w \leftarrow w + \alpha \cdot RMSProp(w, g_w)$
      $w \leftarrow clip(w, -c, c)$
  end for
  - Uzorkovati podskup od m elemenata $\{z_1, ..., z_m\}$ iz raspodele p_z
  - Ažurirati parametre generatora sledećim postupkom:

    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot RMSProp(\theta, g_\theta)$
end while

Vrši se zadati broj iteracija obučavanja diskriminatora po jednoj iteraciji obučavanja generatora. Kod JS divergencije obučavanjem diskriminatora do optimalnog gradijenti postaju nula i nastaje problem nestajućih gradijenata. Kako je Vaserštajnova udaljenost neprekidna i skoro svuda diferencijabilna, gradijenti će biti pouzda-

ni. Ovo stvara mogućnost treniranja diskriminatora do optimalnog i smanjuje verovatnoću nastanka kolapsa modela i problema nestajućih gradijenata. Ipak WGAN nije univerzalno rešenje i ostavlja prostor za poboljšanja.

Jedan od problema Vaserštajnovih GAN modela je prisustvo operacije odsecanja vrednosti težina. Odsecanjem vrednosti težina tako da pripadaju intervalu $[-c, c]$ težine se usmeravaju ka ekstremumima ovog intervala. Pogrešan izbor parametra c može dovesti do pojava eksplodirajućih i nestajućih gradijenata. Nameće se potreba za alternativnim načinom obezbeđivanja Lipšic neprekidnosti funkcija, koji neće prouzrokovati navedene probleme.

WGAN-GP je vrsta GAN modela koja koristi metod kažnjavanja odstupanja gradijenata kako bi postigla 1-Lipšic neprekidnost funkcija. Diferencijabilna funkcija f je 1-Lipšic neprekidna ukoliko su joj gradijenti norme najviše 1 [20]. Ovo se postiže kažnjavanjem modela za norme gradijentnog vektora koje odstupaju od te vrednosti. Kako je potrebno postići 1-Lipšic neprekidnost funkcija $\{f_w\}_{w \in \mathcal{W}}$ iz formule (4.5) sledi da je dovoljno izvršiti samo kažnjavanje diskriminatora. Kako nije moguće izvršiti proveru normi u svim tačkama, dovoljno ih je izračunati na podskupu tačaka nastalih interpolacijom između stvarnih i generisanih podataka [10]. Raspodela podataka dobijenih interpolacijom označava se sa $\mathbb{P}_{\hat{x}}$.

Nova funkcija greške kažnjava odstupanje gradijenata proporcionalno faktoru kažnjavanja λ:

$$L = \underbrace{E_{\tilde{x} \sim p_g}[D(\tilde{x})] - E_{x \sim p_{data}}[D(x)]}_{\text{originalna greška diskriminatora}} + \underbrace{\lambda E_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]}_{\text{kažnjavanje odstupanja gradijenata}}$$

gde $\tilde{x}$ označava generisane podatke iz raspodele p_g, x stvarne podatke iz raspodele p_{data}, a $\hat{x}$ podatke nastale interpolacijom između stvarnih i generisanih podataka.

Na osnovu podataka x i $\tilde{x}$ iz raspodela p_{data} i p_g, interpolisana tačka $\hat{x}$ se dobija na sledeći način:

$$\hat{x} \leftarrow \epsilon x + (1 - \epsilon)\tilde{x}, \quad \epsilon \in U(0, 1) \tag{4.6}$$

Iz formule (4.6) vidimo da se podaci iz $\mathbb{P}_{\hat{x}}$ biraju uniformno sa duži koje spajaju tačke iz p_{data} sa tačkama iz p_g. Kažnjavanje norme gradijenata diskriminatora je potrebno izvršiti za svaku uzorkovanu instancu pojedinačno. Ovaj uslov može biti narušen primenom unutrašnje standardizacije, kako se ona izvršava nad podskupovima podataka [10]. Zbog toga se u praksi iz diskriminatora izbacuju slojevi unutrašnje standardizacije.

Proces obučavanja Vaserštajnovih GAN-ova sa kažnjavanjem odstupanja gradijenata predložen u originalnom radu zadat je narednim algoritmom. Korišćene podrazumevane vrednosti hiperparametara: $\lambda = 10$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$, $n_{critic} = 5$ [10].

Algoritam 3 Algoritam obučavanja WGAN-GP modela

Napomena: *Adam*-optimizacijski algoritam sa hiperparametrima α, β_1 i β_2
Hiperparametri: λ-faktor kažnjavanja odstupanja gradijenta , n_{critic}-broj iteracija diskriminatora po jednoj iteraciji generatora, m-veličina podskupa podataka
Parametri: w-parametri diskriminatora(w_0 inicijalna vrednost), θ-parametri diskriminatora(θ_0 inicijalna vrednost)

while θ ne konvergira **do**
  for $t = 0, ..., n_{critic}$ **do**
    • Uzorkovati podskup od m elemenata $\{z_1, ..., z_m\}$ iz raspodele p_z
    • Uzorkovati podskup od m elemenata $\{x_1, ..., x_m\}$ iz raspodele p_{data}
    for $i = 1, ..., m$ **do**
      • Uzorkovati nasumičan broj $\epsilon \sim U(0, 1)$
      $\tilde{x_i} \leftarrow w + G_\theta(z_i)$
      $\hat{x_i} \leftarrow \epsilon x_i + (1 - \epsilon)\tilde{x_i}$
      $L^{(i)} \leftarrow D_w(\tilde{x_i}) - D_w(x_i) + \lambda(\|\nabla_{\hat{x_i}} D_w(\hat{x_i})\|_2 - 1)^2$
    end for
    $w \leftarrow Adam(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  • Uzorkovati podskup od m elemenata $\{z_1, ..., z_m\}$ iz raspodele p_z
  $\theta \leftarrow Adam(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(z)), \theta, \alpha, \beta_1, \beta_2)$
end while

4.3 Evaluacija modela

Evaluacija modela generativnih suparničkih mreža predstavlja još uvek nerešeno pitanje. Želimo da nove slike budu što je moguće većeg kvaliteta ali i da među generisanim podacima postoji raznolikost koja oslikava celokupnu raspodelu podataka. Kako je raspodela modela implicitno zadata skupom generisanih podataka, postavlja se pitanje kako izvršiti upoređivanje sa raspodelom podataka implicitno zadatom trening skupom. Neke od najpoznatijih metrika su *inception score (IS)* i *Fréchet inception distance (FID)* koje se pri statističkom poređenju podataka oslanjaju na rezultate unapred obučene duboke neuronske mreže.

Duboka konvolutivna mreža *Inception v3* [22] dizajnirana je za problem klasifikacije slika nad skupom podataka ImageNet. Skup podataka ImageNet se sastoji od 1.2 miliona slika u boji, koje pripadaju jednoj od mogućih 1000 klasa [3]. Unapred obučeni InceptionV3 modeli naišli su na široku primenu u oblasti mašinskog učenja. *IS* metrika određuje kvalitet generisanih slika na osnovu rezultata klasifikacije modela InceptionV3 prethodno obučenog nad skupom podataka ImageNet. Poistovećuje se sa ljudskom evaluacijom verodostojnosti slika i meri dve stvari:

- Raznovrsnost slika, odnosno koliki broj klasa je pokriven generisanim slikama
- Kvalitet generisanih slika, odnosno sličnost generisanog objekta sa nekim postojećim

Ukoliko neki od ciljeva nije zadovoljen, rezultat će biti manji broj. Veći IS skor govori da je generativna mreža sposobna da produkuje raznovrsne izlaze visokog kvaliteta. Izračunava se na sledeći način:

$$\text{IS}= exp(E_{x\sim p_g} KL(p(y|x) \parallel p(y)))$$
$$p(y) = \int_x p(y|x)p_g(x)dx$$

KL divergencija meri različitost dve raspodele podataka. Posmatra se razlika između marginalne raspodele klasa p(y) i uslovne raspodele podataka $p(y|x)$ izračunate na osnovu modela InceptionV3. Kako želimo raznovrsnost klasa, idealna marginalna raspodela je uniformna. Poželjno je da slike sadrže jasno određene objekte, samim tim uslovna raspodela je niske entropije. Kada su oba uslova zadovoljena očekuje se visoka KL divergencija i velika IS vrednost [4]. Kako bi se lakše uočio rast KL divergencije, posmatra se eksponent njenog očekivanja po skupu generisanih podataka. IS metrika je ograničena na domen slika i ne može se primeniti za ocenu kvaliteta generisanih podataka drugog tipa.

IS se pokazao neuspešan u otkrivanju situacija pri kojima GAN za određene klase uvek generiše jednu ili par istih instanci. Ovo ograničenje prevazilazi FID, poznat i kao *Wasserstein-2* udaljenost. Stvarni i generisani podaci propuštaju se kroz model InceptionV3 koji izdvaja vizuelno značajne atribute [5]. Za dve Gausove raspodele stvarnih i generisanih atributa zadate prosekom i odstupanjem (m_r, c_r) i (m_g, c_g) FID se izračunava na sledeći način:

$$\text{FID}= \|m_r - m_g\|_2^2 + Tr(c_r + c_g - 2(c_r c_g)^{\frac{1}{2}})$$

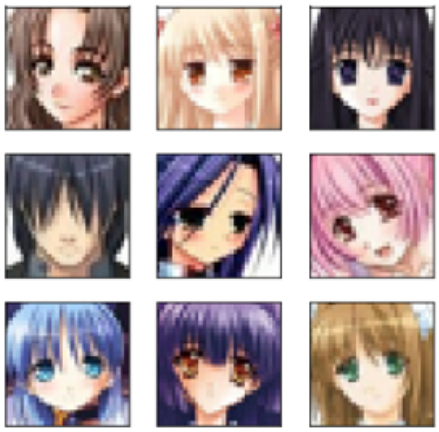
FID se dobro slaže sa ljudskom procenom kvaliteta i raznolikosti slika [26]. Niži FID povlači bolji kvalitet slika.

Izračunavanje IS i FID metrika oslanja se na rezultate modela zasnovanog na skupu podataka ImageNet koji se sastoji od slika određene dimenzije. Samim tim postavlja se pitanje kolika je uspešnost metrika na drugim domenima. Iako nisu sasvim pouzdane, FID i IS su veoma popularne metrike za evaluaciju rezultata generativnih suparničkih mreža. U nastavku će se na praktičnom primeru uočiti problem upoređivanja modela na osnovu grafika funkcije greške i potreba za IS i FID metrikama. Interpretacija gubitaka suprostavljenih modela nije jednostavna. Na osnovu nje se mogu uočiti problemi nastali tokom obučavanja, ali ne obezbeđuje informaciju o kvalitetu i raznovrsnosti generisanih slika.

Glava 5

Eksperimentalni rezultati

U nastavku će na praktičnom primeru biti izvršeno poređenje eksperimentalnih rezultata predloženih rešenja. Kako se generativne suparničke mreže prevashodno koriste za generisanje slika, eksperimenti su sprovodeni u tom domenu. Obučavanje se vrši nad skupom podataka od 10000 slika lica anime likova. Slike su u boji, dimenzija 32x32 piksela i predstavljaju podskup originalnog *Anime face dataset* [6] skupa podataka.



Slika 5.1: Podskup instanci trening skupa

Modeli su trenirani na *Asus TUF GAMING F15 FX506* prenosnom računaru sa 16GB interne memorije, grafičkom karticom *nVidia GeForce RTX 3050* i procesorom *Intel Core i7-11800H @ 2030GHz*. Implementacija rešenja vršena je u programskom jeziku Python. Za definisanje i trening generativnih suparničkih mreža korišćena je biblioteka Keras. Eksperimenti su vršeni pojedinačno u okviru zasebnih Jupyter svesaka. Na osnovu obučenih modela izvršena su poređenja predloženih tehnika.

5.1 Opis implementacije modela

Kako se trening skup sastoji od 10000 slika u boji, generator i diskriminator su implementirani kao duboke konvolutivne neuronske mreže.

Preprocesiranje podataka

Izvršava se postupak normalizacije ulaznog skupa podataka svođenjem vrednosti pojedinačnih piksela na vrednosti iz intervala $[-1, 1]$. Kako poslednji sloj generatora koristi *Tanh* aktivacionu funkciju, opisani postupak omogućava da vrednosti pojedinačnih piksela stvarnih i generisanih slika budu iz istog intervala.

Kreiranje generatora

Biblioteka *Keras* omogućava definisanje arhitekture modela kao sekvence slojeva. Prilikom implementacije poštovane su predložene izmene za postizanje stabilnijeg obučavanja DCGAN modela. Povećanje dimenzije vrši se operacijom transponovane konvolucije koja je podržana kroz `Conv2DTranspose` sloj. Unutrašnja standardizacija je podržana kroz `BatchNormalization` slojeve.

```
Model: "generator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16384)             1654784
_________________________________________________________________
batch_normalization (BatchNo (None, 16384)             65536
_________________________________________________________________
re_lu (ReLU)                 (None, 16384)             0
_________________________________________________________________
reshape (Reshape)            (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 16, 16, 256)       1638656
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 256)       1024
_________________________________________________________________
re_lu_1 (ReLU)               (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 32, 32, 128)       819328
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 128)       512
_________________________________________________________________
re_lu_2 (ReLU)               (None, 32, 32, 128)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 32, 32, 3)         9603
=================================================================
Total params: 4,189,443
Trainable params: 4,155,907
Non-trainable params: 33,536
_________________________________________________________________
```

Slika 5.2: Arhitektura generatora

ReLU aktivaciona funkcija korišćena je u svim slojevima, osim u poslednjem u kom se koristi *Tanh* aktivaciona funkcija. Na primeru sa slike 5.2 prikazana je arhitektura jednog od generatora koji će biti korišćen tokom eksperimenata. Na osnovu vektora nasumičnih vrednosti dimenzije 100 opisani model generiše slike dimenzija $32 \times 32 \times 3$. Da bi to bilo moguće potpuno povezani sloj obezbeđuje $8 \cdot 8 \cdot 256$ izlaza čija dimenzija se potom menja uz pomoć `Reshape` sloja u vektor dimenzije $(8 \times 8) \times 256$. Primenom sloja transponovane konvolucije sa 256 filtera i pomerajem $(2, 2)$ dobija se izlaz dimenzije $(16 \times 16) \times 256$. Narednom primenom sloja transponovane konvolucije sa 128 filtera i pomerajem $(2, 2)$ dobija se izlaz dimenzije $(32 \times 32) \times 128$. Zarad postizanja stabilnijeg obučavanja svaki `Conv2DTranspose` sloj praćen je `BatchNormalization` slojem i *ReLU* aktivacionom funkcijom. Poslednji konvolutivni sloj transformiše izlaz u sliku dimenzije $32 \times 32 \times 3$, dok *Tanh* aktivaciona funkcija svodi vrednosti piksela na interval $[-1, 1]$.

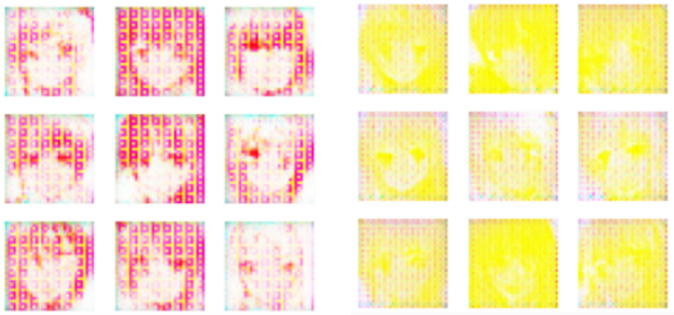
Kreiranje diskriminatora

Diskriminator je jednostavan binarni klasifikator prilikom čije implementacije su poštovane izmene za postizanje stabilnijeg obučavanja DCGAN-ova.

```
Model: "discriminator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32, 32, 1)         4
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 128)       3328
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 256)         819456
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 8, 8, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dropout (Dropout)            (None, 16384)             0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 16385
=================================================================
Total params: 840,709
Trainable params: 839,941
Non-trainable params: 768
_________________________________________________________________
```

Slika 5.3: Arhitektura diskriminatora

Na primeru sa slike 5.3 prikazana je arhitektura jednog od diskriminatora koji će biti korišćen tokom eksperimenata. Ulazni podaci se sastoje od stvarnih i generisanih slika dimenzije $32 \times 32 \times 3$ koje je potrebno klasifikovati. Broj filtera konvolutivnih slojeva raste tokom izgradnje diskriminatora. Prvi konvolutivni sloj sadrži 128 filtera, dok naredni konvolutivni sloj sadrži 256 filtera. To je u suprotnosti sa arhitekturom generatora, gde broj filtera u slojevima transponovane konvolucije opada. Rezultati izvršenih eksperimenata su pokazali da ovakva arhitektura suparničkih mreža obezbeđuje rezultate veće rezolucije u odnosu na arhitekturu gde broj filtera konvolutivnih slojeva diskriminatora opada. Na primeru sa slike 5.4 vidi se razlika između generisanih instanci dva modela koji se razlikuju jedino u opisanom poretku broja filtera konvolutivnih slojeva.



Slika 5.4: Slike generisane nakon 20 epoha obučavanja dva modela sa različitim redosledom navođenja broja filtera konvolutivnih slojeva

Funkcija greške

Funkcija greške diskriminatora mora da prikaže odstupanje njegovih izlaza od željenih vrednosti. Očekivana izlazna vrednost diskriminatora je 1 za stvarne instance i 0 za generisane. Ukupna greška diskriminatora predstavlja zbir greške predviđanja nad stvarnim podacima i greške predviđanja nad generisanim podacima. Opisane greške se izračunavaju funkcijom binarne unakrsne entropije. Ukoliko vektori `real` i `fake` predstavljaju predviđanja diskriminatora nad stvarnim i generisanim podacima redom, funkcija greške diskriminatora implementirana je na sledeći način:

```
bloss=tf.keras.losses.BinaryCrossentropy()
def discriminator_loss(real,fake):
    real_loss=bloss(tf.ones_like(real),real)
    fake_loss=bloss(tf.zeros_like(fake),fake)
    return real_loss + fake_loss
```

Funkcijom binarne unakrsne entropije meri se odstupanje vektora `real` od vektora čije su sve vrednosti jednake jedan. Takav vektor nazivamo vektorom pozitivnih instanci. Analogno, vektor čije su sve vrednosti 0 nazivamo vektor negativnih instanci.

Cilj generatora je da uspešno prevari diskriminator da pogrešno klasifikuje generisane podatke kao stvarne. Zbog toga funkcija greške generatora meri odstupanje predviđanja diskriminatora nad generisanim podacima od vektora pozitivnih instanci i implementirana je na sledeći način:

```
def generator_loss(fake):
    g_loss=bloss(tf.ones_like(fake),fake)
    return g_loss
```

U slučaju **jednostranog skaliranja ciljnih promenljivih** vrednosti pozitivnih instanci umanjuju se za mali pozitivan faktor. Ukoliko je vrednost hiperparametra `FACTOR` 0.1 potrebno je izvršiti sledeću transformaciju vektora pozitivnih instanci:

$$[1, 1, 1, ..., 1, 1] \rightarrow [0.9, 0.9, 0.9, ..., 0.9, 0.9]$$

Implementacija navedene tehnike zahteva izmenu funkcije greške diskriminatora, odnosno dela u kom se izračunava gubitak na osnovu stvarnih instanci:

```
real_input=tf.ones_like(real)
real_input *= (1 - FACTOR)
real_loss=bloss(real_input,real)
```

U slučaju **WGAN** i **WGAN-GP** modela zadatak diskriminatora je da proceni koliko verodostojno izgledaju podaci. Cilj obučavanja diskriminatora je povećanje razlike između vrednosti koje dodeljuje stvarnim i generisanim instancama. Ukoliko vektori `real` i `fake` predstavljaju vrednosti koje diskriminator dodeljuje stvarnim i generisanim instancama redom, funkcije greške generatora i diskriminatora su implementirane na sledeći način:

```
def discriminator_loss(real,fake):
    return tf.reduce_mean(fake) - tf.reduce_mean(real)

def generator_loss(fake):
    return -tf.reduce_mean(fake)
```

Uz pomoć *TensorFlow* funkcije `tf.reduce_mean` izračunavaju se proseci prosleđenih vektora vrednosti `real` i `fake`.

Obučavanja modela

U nastavku su opisani parametri i hiperparametri korišćeni prilikom izvršenih eksperimenata. Ukratko, bez zalaženja u detalje, opisani su specifični delovi implementacija posmatranih tehnika.

Proces obučavanja modela definisan je funkcijom `train_model`. U posmatranoj funkciji vrši se zadati broj trening epoha i za svaku epohu ispisuju se vrednosti funkcija greške generatora i diskriminatora i tačnost predviđanja diskriminatora nad stvarnim i generisanim podacima. Nakon svake trening epohe prati se kvalitet generisanih slika uz pomoć definisane funkcije `plot_generated`. Za svaku epohu i svaki podskup podataka trening skupa (eng. *batch*) izvršava se jedan prolaz obučavanja.

Prilikom implementacije **tehnike dodavanja šuma** podskupovima podataka trening skupa se prethodno dodaje šum iz uniformne $U(-1, 1)$ raspodele regulisan multiplikativnim hiperparametrom `NOISE_FACTOR` za određivanje jačine šuma.

Biblioteka *TensorFlow*, kroz `GradientTape` API, obezbeđuje mehanizam za automatsko izračunavanje gradijenata. Na osnovu greške modela, pozivom metoda `gradient` nad `GradientTape` objektom, vrši se računanje gradijenata. Izračunati gradijenti se koriste za izmenu težina parametara modela i primenjuju se pozivom metoda `apply_gradients` nad odgovarajućim optimizatorom.

Generator i diskriminator koriste *Adam* optimizator prilikom implementacije GAN i WGAN-GP modela. U zavisnosti od eksperimenata koriste se različite vrednosti hiperparametara `learning_rate` i `beta_1` zadatih optimizatora. Implementacija WGAN modela koristi *RMSProp* optimizator, kako korišćenje *Adam* optimizatora dovodi do nestabilnosti modela [2]. U slučaju **WGAN** modela, hiperparametrom `EXTRA_STEPS` vrši se kontrolisanje broja iteracija obučavanja diskriminatora po jednoj iteraciji obučavanja generatora. Odsecanje težina WGAN modela zadaje se kao objekat ograničenja koji se prosleđuje konvolutivnim `Conv2D` slojevima diskriminatora kroz parametar `kernel_constraint`. Ograničenje je definisano kao klasa koja nasleđuje `keras.constraints.Constraint` klasu i predefinisani `__call__()` metod vrši odsecanje težina na osnovu prosleđenog parametra odsecanja.

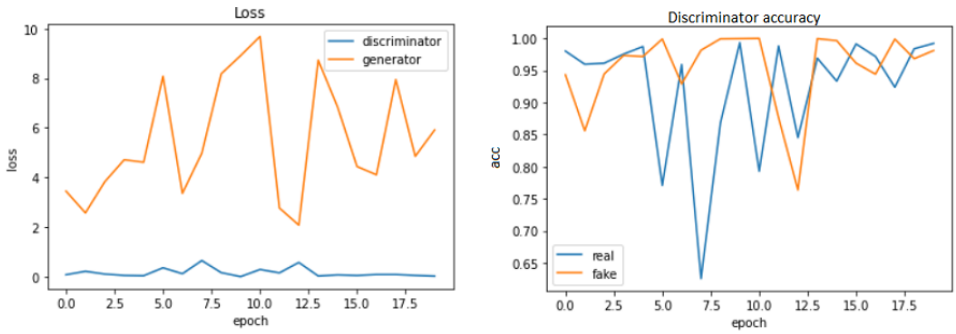
Pored hiperparametra `EXTRA_STEPS` **WGAN-GP** modeli uvode hiperparametar `GP_WEIGHT` koji kontroliše jačinu kažnjavanja odstupanja gradijenata. Kazna se izračunava na osnovu slika dobijenih interpolacijom i pomnožena vrednošću `GP_WEIGHT` se dodaje izračunatoj vrednosti greške diskriminatora.

5.2 Poređenje rezultata

Na praktičnim primerima izvršena su poređenja rezultata opisanih metoda. U nastavku je prikazan jedan od eksperimenata, prilikom kog su korišćeni:

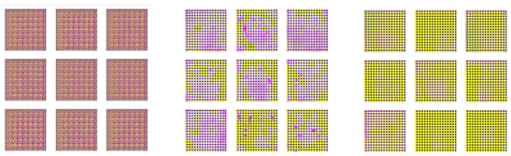
- Generator i diskriminator prikazani na slikama 5.2 i 5.3
- Adam optimizator sa korakom učenja 0.0002 i parametrom $\beta_1 = 0.5$
- Veličina podskupa za treniranje $BATCH\_SIZE = 100$
- Dimenzija latentnog prostora generatora, odnosno dužina vektora slučajnih vrednosti $LAT\_DIM = 100$

Rezultati su praćeni tokom 20 trening epoha. Na slici 5.5 su prikazani grafik funkcije greške generatora i diskriminatora i grafik funkcije tačnosti predviđanja diskriminatora nad stvarnim i generisanim podacima tokom obučavanja.



Slika 5.5: Praćenje funkcija greške i tačnosti predviđanja diskriminatora

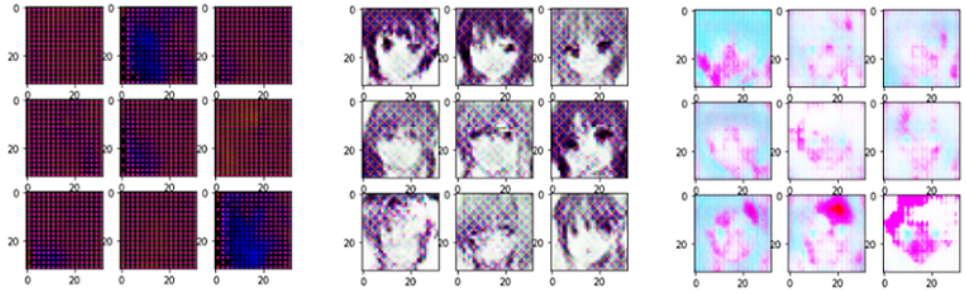
Sa grafika funkcija greške uočava se da je vrednost gubitka diskriminatora približno jednaka nuli tokom obučavanja. Gubitak generatora varira i dostiže visoke vrednosti. Diskriminator sa velikom verovatnoćom uspešno klasifikuje stvarne i generisane podatake. Samim tim usled preprilagođavanja ne obezbeđuje korisnu povratnu informaciju i onemogućava dalji napredak generatora. Na osnovu generisanih slika i gubitka modela može se naslutiti kolaps modela u ranim fazama obučavanja.



Slika 5.6: Generisane slike nakon 1, 10 i 20 trening epoha redom

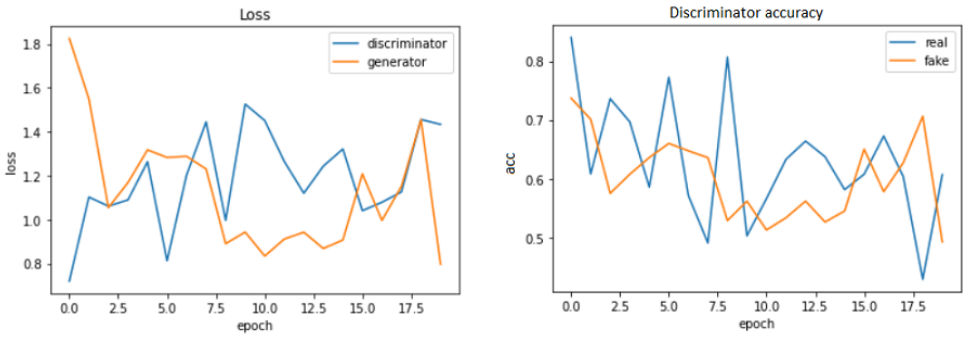
Jednostrano skaliranje ciljnih promenljivih

Posmatrane su različite vrednost hiperparametra `FACTOR` koji smanjuje sigurnost sa kojom diskriminator tvrdi da je neka instanca stvarna. Rezultati generisani nakon 20 trening epoha prikazani su na slici 5.7.



Slika 5.7: Generisane slike za vrednosti 0.1, 0.15 i 0.2 hiperparametra `FACTOR`

Male vrednosti hiperparametra, kao `FACTOR` = 0.1, dovode do sličnih rezultata kao polazni model. Diskriminator uspešno klasifikuje podatke i vrednost funkcije gubitka je približna nuli. Vrednost hiperparametra `FACTOR` = 0.15 sprečava preprilagođavanje diskriminatora. Slika 5.8 prikazuje posmatrane grafike funkcije greške i tačnosti predviđanja diskriminatora, tokom 20 trening epoha. Vrednosti tačnosti predviđanja diskriminatora variraju sa prosekom 0.6. Funkcija greške diskriminatora raste, dok funkcija greške generatora opada. Opisano željeno ponašanje uočava se i u eksperimentima sa većim brojem trening epoha.

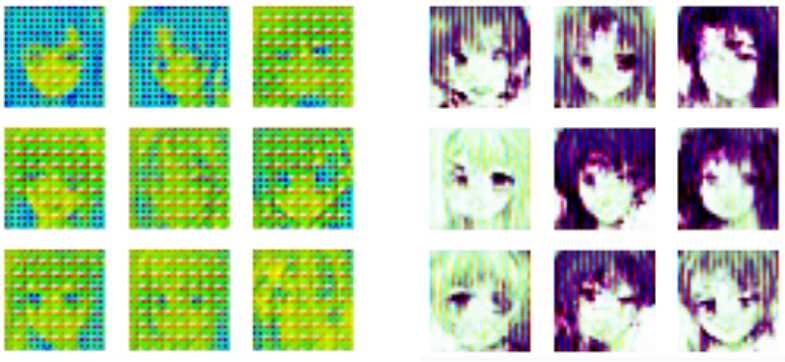


Slika 5.8: Funkcija greške i tačnost predviđanja diskriminatora

Velike vrednosti hiperparametra `FACTOR` smanjuju značaj povratne informacije od strane diskriminatora i dovode do pada kvaliteta generisanih slika. Primer su slike generisane za vrednost `FACTOR` = 0.2.

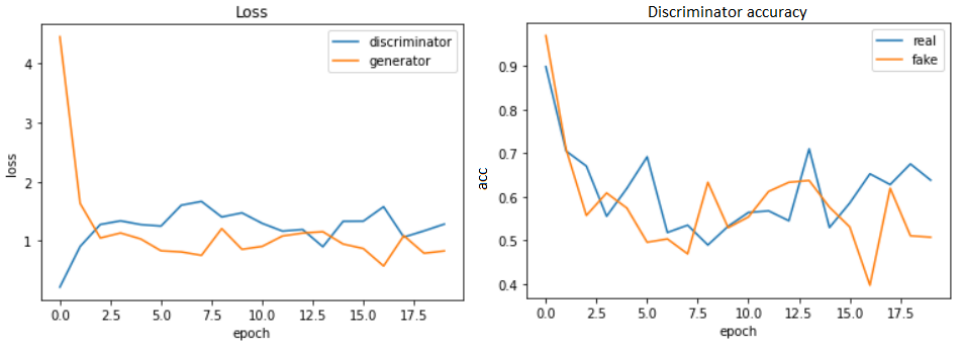
Dodavanje šuma

Jačina dodatog nasumičnog šuma regulisana je hiperparametrom `NOISE_FACTOR`. Na slici 5.9 su prikazane generisane slike nakon 20 trening epoha za vrednosti `NOISE_FACTOR` = 0.009 i `NOISE_FACTOR` = 0.0005.



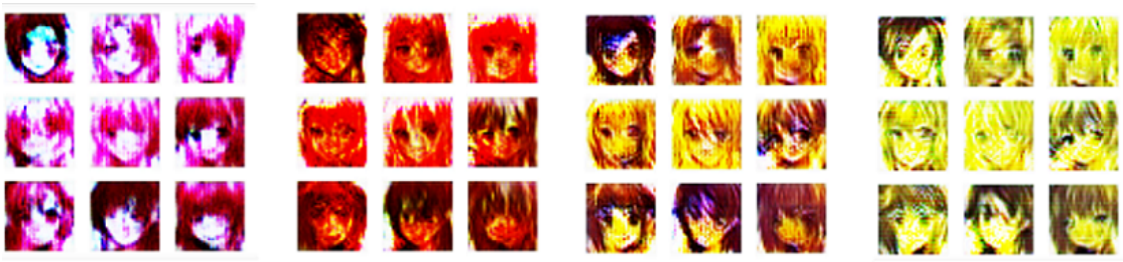
Slika 5.9: Generisane slike za vrednosti 0.009 i 0.0005 hiperparametra `NOISE_FACTOR`

Male vrednosti hiperparametra nemaju dovoljan uticaj na diskriminator. Velike vrednosti mogu dovesti do promene strukture generisanih slika i pojave šuma na njima. Ovo je uočljivo na generisanim slikama za vrednost `NOISE_FACTOR` = 0.009. Kao najbolja vrednost prilikom eksperimenata pokazala se vrednost 0.0005.



Slika 5.10: Praćenje obučavanja za vrednost `NOISE_FACTOR` = 0.0005

Sa slike 5.10 uočava se da još u ranim fazama obučavanja dolazi do stabilizacije funkcija greške generatora i diskriminatora. Prilikom dodatnog eksperimenta od 50 epoha, na osnovu posmatranih generisanih slika, uočen je delimični kolaps modela. Prikaz generisanih slika tokom obučavanja dat je na slici 5.11.



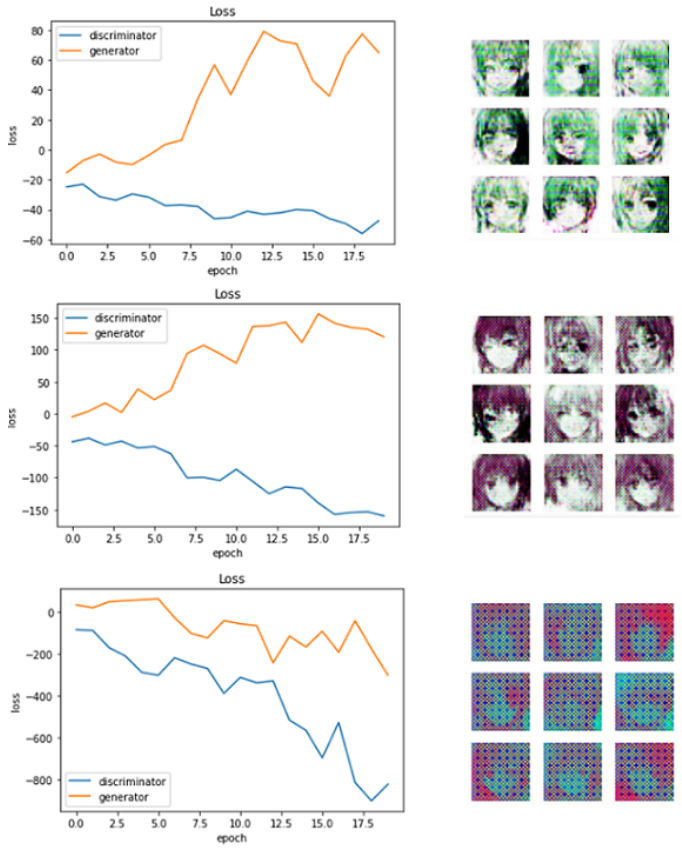
Slika 5.11: Generisane slike nakon 10, 25, 40 i 50 trening epoha

WGAN

U odnosu na početni GAN model uvode se sledeće razlike:

- RMSprop optimizacioni metod sa korakom učenja 0.00005
- Faktor odsecanja težina 0.01

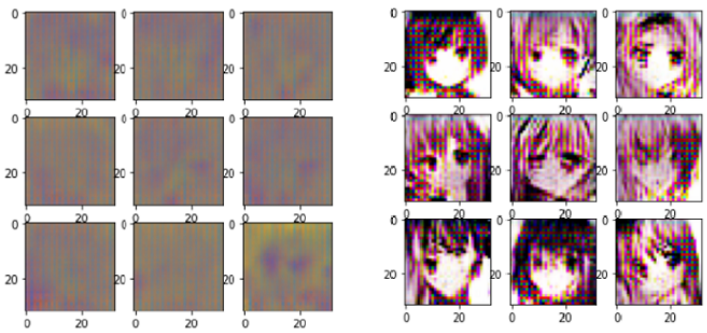
Tokom eksperimenata koriste se vrednosti 2, 3 i 4 hiperparametra `EXTRA_STEPS`. Slika 5.12 prikazuje grafike funkcija greške i generisane slike nakon 20 trening epoha.



Slika 5.12: Rezultati obučavanja za vrednosti 2, 3 i 4 hiperparametra `EXTRA_STEPS`

Implementacija funkcija `discriminator_loss` i `generator_loss` kod WGAN modela zahteva novi način interpretacije dobijenih grafika. Povećanjem vrednosti hiperparametra `EXTRA_STEPS` povećava se broj iteracija obučavanja diskriminatora po jednoj iteraciji generatora. Vidi se da funkcija greške strmije opada i da razlika vrednosti koje diskriminator dodeljuje stvarnim i generisanim slikama postaje sve veća. Ovo rezultuje poboljšanjem kvaliteta generisanih slika promenom vrednosti hiperparametra sa 2 na 3. Ipak dalje povećanje vrednosti hiperparametra dovodi do nezadovoljavajućih rezultata u prvih 20 trening epoha.

Problem nestajućih gradijenata i kolapsa modela uočen je na više posmatranih DCGAN modela. Pogrešan izbor faktor odsecanja težina može dovesti do pojave eksplodirajućih i nestajućih gradijenata. Ipak, primena Vaserštajnove funkcije greške i vrednosti `EXTRA_STEPS=3` često se pokazala uspešnom prilikom rešavanja navedenih problema, posebno kolapsa modela. Jedan od primera rezultata dobijenih od strane DCGAN i odgovarajućeg WGAN modela prikazan je na slici 5.13.



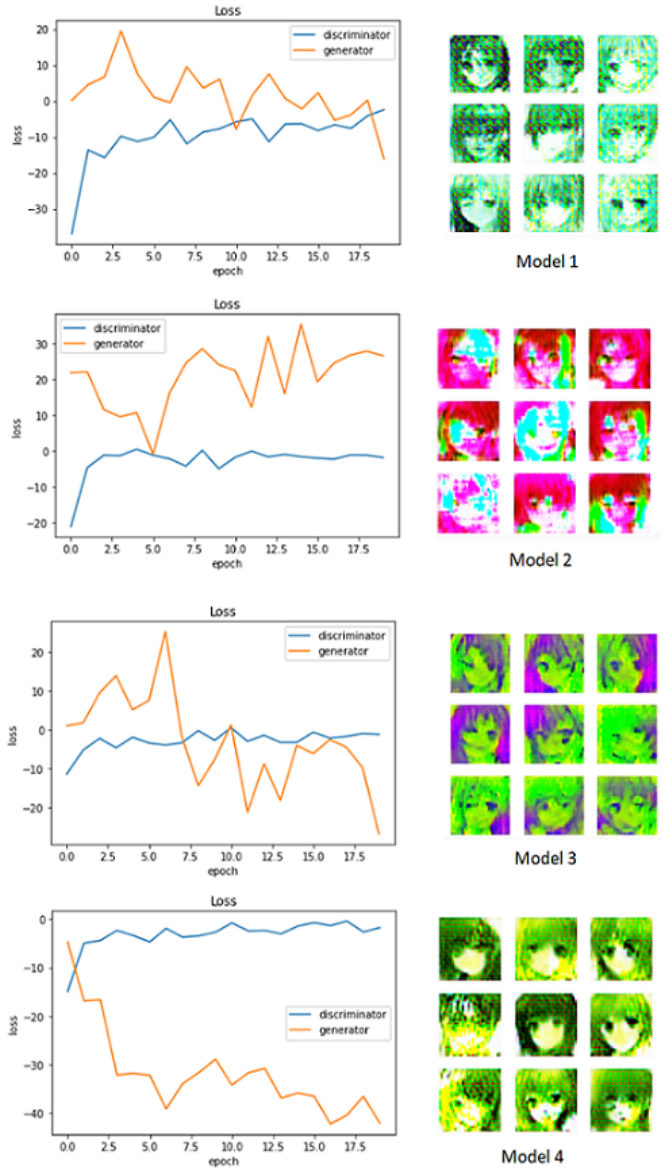
Slika 5.13: Slike dobijene nakon 60 epoha obučavanja DCGAN i WGAN modela

WGAN-GP

Za razliku od `BatchNormalization` slojeva, uz pomoć `LayerNormalization` slojeva standardizacija podataka se vrši nad svakom instancom iz podskupa pojedinačno. Kako se kažnjavanje odstupanja gradijenata diskriminatora vrši za svaku instancu pojedinačno [10], urađeno je poređenje narednih modela:

- Model 1 - `BatchNormalization` slojevi prisutni u generatoru i diskriminatoru
- Model 2 - odsustvo slojeva standardizacije u strukturi diskriminatora
- Model 3 - odsustvo slojeva standardizacije u generatoru i diskriminatoru
- Model 4 - zamena `BatchNormalization` slojeva `LayerNormalization` slojevima u strukturi diskriminatora

Korišćene su sledeće vrednosti hiperparametara: `EXTRA_STEPS=3`, `GP_WEIGHT=10.0`. Slika 5.14 prikazuje rezultate nakon 20 epoha obučavanja modela.



Slika 5.14: Rezultati WGAN-GP modela

Interesantno je da uklanjanje slojeva standardizacije iz oba modela dovodi do boljih rezultata od uklanjanja slojeva standardizacije jedino na strani diskriminatora. Ipak, u oba slučaja izbacivanje `BatchNormalization` slojeva ima negativan uticaj i dovodi do nestabilnog procesa obučavanja. Zamena `BatchNormalization` slojeva sa `LayerNormalization` slojevima doprinosi poboljšanju kvaliteta generisanih slika.

Poređenje modela

Kako se generativne suparničke mreže zasnivaju na ideji suprostavljenih neuronskih mreža, interpretacija grafika funkcije greške predstavlja izuzetno težak zadatak. Iz posmatranih primera može se uočiti da se na osnovu funkcije gubitka ne može oceniti raznovrsnost i kvalitet generisanih slika. Periodično uzorkovanje slike ne daje dovoljno informacija o raznolikosti generisanih podataka. Zato je potrebno izvršiti dodatnu evaluaciju dobijenih modela na osnovu odgovarajućih metrika.

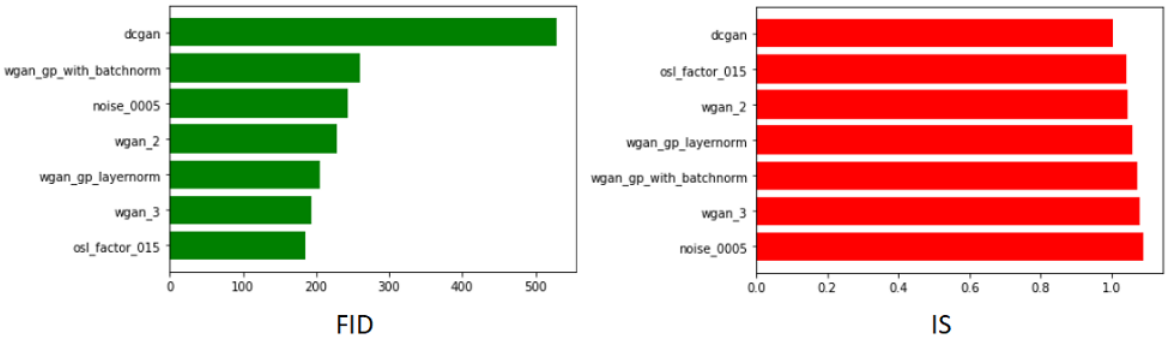
Na osnovu kvaliteta uzorkovanih slika, za svaku od opisanih tehnika izabrani su najbolji modeli za koje je izvršeno dodatno izračunavanje FID i IS vrednosti. U nastavku su prikazani odabrani modeli i vrednosti izračunatih metrika zaokružene na tri decimale.

Odabrani model	**IS**	**FID**
DCGAN bazni model	1.004	528.781
Jednostrano skaliranje ciljnih promenljivih `FACTOR=0.15`	1.039	184.859
Dodavanje šuma `FACTOR=0.0005`	1.089	242.853
WGAN `EXTRA_STEPS=2`	1.043	228.314
WGAN `EXTRA_STEPS=3`	1.080	193.186
WGAN-GP `BatchNormalization`	1.073	260.632
WGAN-GP `LayerNormalization`	1.059	205.009

Tabela 5.1: Rezultati evaluacije modela

Na posmatranom modelu uočava se jedan od problema evaluacije generativnih suparničkih mreža. Izračunavanje zadatih metrika oslanja se na rezultate unapred obučenog modela *Inception V3* zasnovanog na ImageNet skupu podataka i zahteva promenu dimenzija ulaznih slika. Kako se posmatrani skup podataka sastoji od slika anime lica, očekivane su male razlike između dobijenih vrednosti IS metrike. FID meri neslaganje između stvarnih i generisanih podataka i samim tim predstavlja adekvatniju metriku u odnosu na IS. Kao što je već objašnjeno, GAN model se

smatra boljim ukoliko ima manji FID i veći IS. Evaluacija posmatranih modela je izvršena na osnovu 3000 generisanih i 3000 nasumičnih slika stvarnih anima lica. Slika 5.15 prikazuje rezultate evaluacije modela.



Slika 5.15: Evaluacija modela

Iz tabele 5.1 uočava se da najmanji IS i najveći FID ima polazni DCGAN model. Kako je obučavanje modela trajalo 20 epoha, očekivane su visoke FID i niske IS vrednosti. Najmanji FID imaju tehnika jednostranog skaliranja ciljnih promenljivih (184.859) i primena WGAN modela (193.186). Dobijene vrednosti predstavljaju značajno poboljšanje u odnosu na početni model. Iako su uzorkovani generisani rezultati vizuelno zadovoljavajući, najveći FID ima tehnika korišćenja WGAN-GP modela sa `BatchNormalization` slojevima.

Navedeni primer prikazuje izazove obučavanja generativnih suparničkih mreža, od prepoznavanja nastalog problema do problema evaluacije dobijenih modela. Sa slike 5.15 se uočava da predložena tehnika jednostranog skaliranja ciljnih promenljivih uspešno sprečava preprilagođavanje diskriminatora uočeno na grafiku 5.5. Osim toga sprovedeni eksperimenti su pokazali da korišćenje Vaserštajnove funkcije greške sprečava nastanak opisanih problema, ako je izvršen odgovarajući izbor vrednosti hiperparametara.

Glava 6

Zaključak

Generativne suparničke mreže su jedan od najpopularnijih generativnih modela mašinskog učenja. Ipak, njihova primena i razvoj otežani su nestabilnim procesom obučavanja i nedostatakom odgovarajućih metrika za evaluaciju. U ovom radu predstavljen je pregled teorije generativnih suparničkih mreža. Opisani su najčešći problemi koji mogu nastati tokom njihovog obučavanja, kao što su kolaps modela, problem nestajućih gradijenata i problem konvergencije. Za svaki od problema predložena su rešenja koja vode stabilnijem procesu obučavanja. Na praktičnom primeru posmatrano je poređenje rezultata implementiranih rešenja. Evaluacija modela izvršena je uz pomoć *inception score* i *Fréchet inception distance* metrika koje se poistovećuju sa ljudskom evaluacijom verodostojnosti slika. Obrađene su tehnika jednostranog skaliranja ciljnih promenljivih i tehnika dodavanja šuma. Osim funkcije greške zasnovane na binarnoj unakrsnoj entropiji posmatrana je i funkcija greške zasnovana na Vaserštajnovoj udaljenosti, sa i bez odsecanja gradijenata.

Modeli sa Vaserštajnovom funkcijom greške su se veoma uspešno pokazali prilikom rešavanja problema nestajućih gradijenata i kolapsa modela. Doveli su do stabilnijeg procesa obučavanja i kvalitetnijih generisanih slika, bez povećanja broja konvolutivnih slojeva i izmene strukture neuronskih mreža. Uspešnosti WGAN-GP modela zavisi od izbora slojeva standardizacije. Sa druge strane uočeno je da jednostavne tehnike, kao što je jednostrano sklairanje ciljnih promenljivih, mogu dovesti do dovoljno dobrih rezultata. U zavisnosti od problema mogu sprečiti preprilagođavanje diskriminatora bez potrebe za uvođenjem nove funkcije greške.

Rad daje uvid u potencijalne probleme i njihova rešenja, stvarajući prostor za dalji napredak i pronalaženje novih tehnika za postizanje stabilnog obučavanja generativnih suparničkih mreža.

Bibliografija

[1] Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 291–301. PMLR, 2019.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN, 2017. arXiv:1701.07875.

[3] Shane Barratt and Rishi Sharma. A Note on the Inception Score, 2018. arXiv:1801.01973.

[4] Eyal Betzalel, Coby Penso, Aviv Navon, and Ethan Fetaya. A Study on the Evaluation of Generative Models, 2022. arXiv:2206.10935.

[5] Min Jin Chong and David Forsyth. Effectively Unbiased FID and Inception Score and Where to Find Them. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6069–6078, 2020.

[6] Spencer Churchill and Brian Chao. Anime Face Dataset, 2019. `https://www.kaggle.com/ds/379764`.

[7] Ghosh, Anirudha and Sufian, A. and Sultana, Farhana and Chakrabarti, Amlan and De, Debashis. Fundamental Concepts of Convolutional Neural Network. In *Recent Trends and Advances in Artificial Intelligence and Internet of Things*, pages 519–567. 2020.

[8] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks, 2017. arXiv:1701.00160.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, 2014. arXiv:1406.2661.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs, 2017. arXiv:1704.00028.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[12] Andreas Holzinger, Anna Saranti, and Heimo Mueller. An experimental exploration environment for Pattern Analysis and Machine Intelligence, 2021. arXiv:2103.00519.

[13] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015. arXiv:1502.03167.

[14] Predrag Janicic and Mladen Nikolic. *Veštačka inteligencija.* Matematicki fakultet Univerziteta u Beogradu: Beograd, Serbian, 2021. on-line at: `http://poincare.matf.bg.ac.rs/~janicic//books/VI_B5.pdf`.

[15] Bernhard Mehlig. *Machine Learning with Neural Networks.* Cambridge University Press, oct 2021.

[16] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction.* MIT Press, 2022.

[17] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image Colorization Using Generative Adversarial Networks. In *Articulated Motion and Deformable Objects*, pages 85–94. Springer International Publishing, 2018.

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2016. arXiv:1511.06434.

[19] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs, 2016. arXiv:1606.03498.

[20] Rikli Samuel, Bigler Daniel Nico, Pfenninger Moritz, and Osterrieder Joerg. Wasserstein GAN: Deep Generation applied on Bitcoins financial time series, 2021. arXiv:2107.06008.

[21] Rajhans Singh, Pavan Turaga, Suren Jayasuriya, Ravi Garg, and Martin Braun. Non-Parametric Priors For Generative Adversarial Networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5838–5847. PMLR, 2019.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions, 2014. arXiv:1409.4842.

[23] Lilian Weng. From GAN to WGAN, 2019. arXiv:1904.08994.

[24] Maciej Wiatrak, Stefano V. Albrecht, and Andrew Nystrom. Stabilizing Generative Adversarial Networks: A Survey, 2020. arXiv:1910.00927.

[25] Sitao Xiang and Hao Li. On the Effects of Batch and Weight Normalization in Generative Adversarial Networks, 2017. arXiv:1704.03971.

[26] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks, 2018. arXiv:1806.07755.

Biografija autora

Nikolina Kuprešanin rođena je 14.05.1998. u Zrenjaninu. Osnovnu školu „Đura Jakšić" u Perlezu i prirodno-matematički smer Zrenjaninske gimnazije završila je kao učenik generacije i nosilac Vukove diplome. Smer Računarstvo i informatika na Matematičkom fakultetu upisuje 2017. godine. Diplomirala je u roku 2021. godine sa prosekom 9.83. Obrazovanje nastavlja na istom fakultetu gde 2021. godine upisuje master akademske studije. Dobitnik je nagrade „Nedeljko Parezanović" Katedre za računarstvo i informatiku, nagrade Matematičkog fakulteta za najbolje studente i stipendije „Dositeja" Fonda za mlade talente Republike Srbije.

Tokom akademske 2021/22 godine zaposlena je kao saradnik u nastavi na Katedri za računarstvo i informatiku Matematičkog fakulteta. Drži vežbe iz kurseva „Uvod u relacione baze podataka", „Kompilacija programskih jezika" i „Računarske mreže". Od 2022. godine započinje karijeru kao softverski inženjer. Trenutno je zaposlena u kompaniji *FIS (Fidelity Information Services)*.