



Text To Image Generation StackGAN

Računarska inteligencija

Autor: Maša Cucić

26. septembar 2024.

Contents

1	Uvod	2
2	Skup podataka	3
3	Metodologija	4
3.1	StackGAN	4
3.2	Conditioning Augmentation Network	5
3.3	Stage 1	5
3.4	Stage 2	6
3.4.1	Greška generatora	6
3.4.2	Greška diskriminatora	7
3.5	Trening	8
3.5.1	Trening nad jednom vrstom ptica	9
3.5.2	Trening nad 40 vrsti ptica	10
4	Test	11
5	Rezultati i zaključak	12

1 Uvod

Tema ovog rada je generisanje slika na osnovu tekstualnih opisa, sa fokusom na kreiranje slika ptica.

Postoji nekoliko pristupa za rešavanje ovog problema korišćenjem GAN mreža. Generative Adversarial Network (GAN) je model dubokog učenja koji se sastoji od dva suprotstavljenih modela, generatora i diskriminatora. Generator i diskriminator se zajedno treniraju. Cilj generatora je da nauči da stvara realističke podatke, dok diskriminator pokušava da razlikuje lažne podatke (generisane od strane generatora) od stvarnih podataka. Ova dva modela se na taj način međusobno unapređuju.

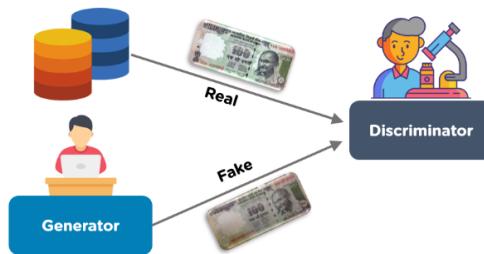


Figure 1: GAN

Rad je zasnovan na StackGAN pristupu. To je napredniji pristup od tradicionalnih GAN-ova, kod kojeg postoji slojevita arhitektura. Trening se vrši u dve faze. U okviru svake od faza imamo generator i diskriminator. U prvoj fazi se generišu slike niske rezolucije, koje se zatim unapređuju ka višoj rezoluciji u narednoj fazi. Detalji o implementaciji StackGAN-a biće obrađeni u daljem tekstu.

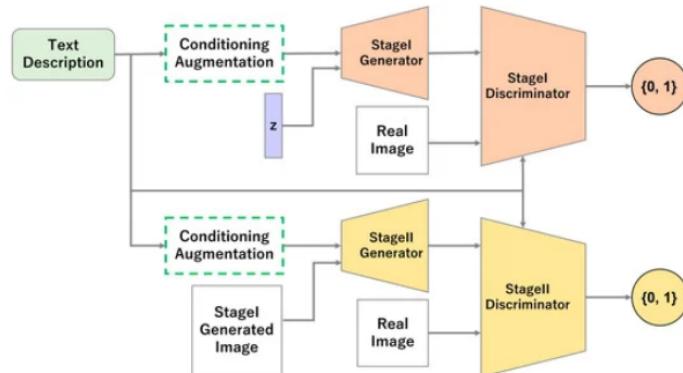


Figure 2: StackGAN

2 Skup podataka

Za obuku modela korišćen je skup podataka CUB-200-2011 (Caltech-UCSD Birds-200-2011). Skup podataka je preuzet sa linka koji se nalazi u referencama [1]. Ovaj skup obuhvata 200 različitih vrsta ptica i sadrži preko 11000 slika. Zbog tehničkih ograničenja, veličina skupa i odabir određenih vrsta je varirao tokom rada na projektu. Najveći fokus bio je na treningu 40 vrsti ptica, kao i na treningu samo jedne vrste.

Podela na trening i test skup je već odraćena, preuzimanjem odgovarajućih embedding-a odnosno tekstualnih opisa. U folderima train i test, nalaze se po tri fajla:

- char-CNN-RNN_embeddings.pickle - predstavlja vektorske reprezentacije opisa slika
- class_info.pickle - sadrži informacije o klasama za svaku sliku
- filenames.pickle - sadrži listu svih naziva fajlova slika

Preuzimanjem ovih fajlova, automatski je završena podela na trening i test skup jer su opisani fajlovi preuzeti dvostruko, i za trening i za test skup. Odnos između trening skupa i test skupa iznosi 75:25.

Što se tiče samog skupa podataka, sastoji se iz 200 različitih vrsti ptica. Za svaku vrstu, pridruženo je oko 60 slika ptica te vrste, kao i 10 tekstualnih opisa za svaku od slika.

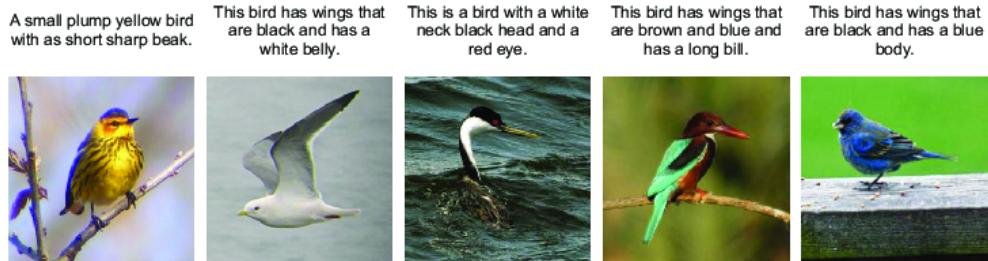


Figure 3: CUB_200_2011

3 Metodologija

3.1 StackGAN

StackGAN (Stacked Generative Adversarial Networks) je napredni pristup u generativnim adversarialnim mrežama koji koristi slojevitu strukturu za generaciju slika iz tekstualnih opisa. Imamo dve glavne faze u kojima se vrši generacija i procenjivanje kvaliteta slika.

- Stage 1
- Stage 2

Obe faze imaju definisane svoj generator i svoj diskriminator. U prvoj fazi, generator kreira slike niske rezolucije (64×64) koje odražavaju osnovne karakteristike tekstualnih opisa, dok diskriminator procenjuje te slike, razlikujući stvarne od generisanih.

U drugoj fazi, generator unapređuje slike dobijene iz prve faze, kako bi generisao slike više rezolucije (256×256), dodajući dodatne detalje slikama. Diskriminator druge faze procenjuje kvalitet ovih poboljšanih slika, slično kao što to radi diskriminator u prvoj fazi.

Tokom treninga, generator i diskriminator se „takmiče“: generator pokušava da stvara slike što sličnije stvarnim, dok diskriminator nastoji da poboljša svoju sposobnost da razlikuje stvarne slike od generisanih. Ovo međusobno takmičenje omogućava modelima da se stalno unapređuju, kako bi generisane slike na kraju bile dobrog kvaliteta.

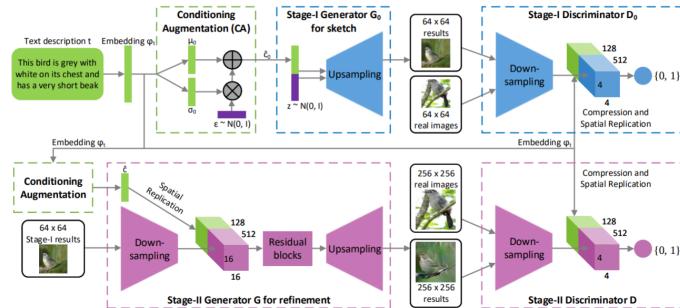


Figure 4: StackGAN

3.2 Conditioning Augmentation Network

Conditioning Augmentation Network (CANet) u StackGAN modelu igra ključnu ulogu u generisanju slika ptica visokog kvaliteta na osnovu tekstualnih opisa. CANet transformiše tekstualne embedding-e (opise) u uslovne vektore koje generativne mreže (generator i diskriminator) mogu efikasnije iskoristiti za generisanje slika na osnovu tih opisa.

Conditioning Augmentation je tehnika koja se koristi kako bi se povećala raznovrsnost uslovnih varijabli koje se koriste za generisanje podataka. Umesto da se oslanja na fiksne uslove (npr. tekstualne embedding-e), ova tehnika nasumično uzima latentne varijable (c) iz nezavisne normalne raspodele. Cilj je da se poveća raznovrsnost tokom obuke i poboljša otpornost modela na male promene u uslovima.

Prvo, kroz funkciju encode, text embedding se projektuje u viši dimenzionalni prostor pomoću linearne transformacije. Rezultat ove projekcije deli se na dva dela: mu (srednja vrednost) i logvar (logaritamska varijansa). Srednja vrednost predstavlja centralnu tačku distribucije mogućih vektora, dok logaritamska varijansa određuje širinu te distribucije, tj. koliko daleko vrednosti mogu da variraju oko srednje vrednosti.

Zatim se, pomoću reparametrizacione funkcije, generiše uslovni vektor c , koji predstavlja nasumičnu varijaciju kreiranu kombinovanjem srednje vrednosti i šuma skaliranog standardnom devijacijom. Na kraju, mreža vraća tri vrednosti: uslovni vektor (c), srednju vrednost (mu), i logaritamsku varijansu (logvar), što omogućava modelu da generiše raznovrsnije slike na osnovu istih tekstualnih opisa.

3.3 Stage 1

U Stage 1 fazi, glavni zadatak je da se iz tekstualnih opisa kreiraju slike niske rezolucije koje prikazuju osnovne karakteristike ptica. Generator koristi tekstualne opise da stvori slike koje prikazuju osnovne oblike i boje ptica, ali bez mnogo detalja.

Ulas za generator predstavljaju:

- **text_embedding** - odgovarajući tekstualni opis na osnovu koga generator kreira slike
- **noise** - slučajni šum definisan pomoću z_dim , koja predstavlja dimenziju latentnog prostora

Najpre, tekstualni opis se pretvara u nasumični uslovni vektor pomoću CA mreže, koji se zatim kombinuje sa prosleđenim šumom (noise). Generator prolazi kroz nekoliko slojeva, uključujući upSamplingBlock, koji

koristi upsampling metodu zasnovanu na nearest neighbors za povećanje dimenzija generisane slike, a zatim primenjuje konvolutivne slojeve definisane sa conv3x3 kako bi poboljšao kvalitet i detalje slike. Generator vraća generisanu sliku, kao i srednju vrednost (μ) i varijansu ($\log \sigma^2$) latentnog prostora, dobijene iz CA mreže, koje će kasnije biti potrebne.

Sa druge strane, diskriminator na ulazu prima:

- **condition** - uslov
- **slika** - slika na koju se taj uslov odnosi

Za dobijenu sliku, diskriminator treba da proceni da li je prava ili lažna. Kao condition se kasnije prosledjuje latentna varijabla μ . Diskriminatore u ovoj fazi procenjuje kvalitet tih slika, koristeći složenu arhitekturu konvolutivnih slojeva kako bi odlučio da li je slika realna ili lažna. On kodira datu sliku, obrađuje uslov, i spaja sliku i uslov u jednu varijablu, na osnovu koje se određuje izlaz diskriminatora za datu sliku i uslov. Diskriminator vraća verovatnoću da je data slika lažna odnosno prava. Vrednost bliža nuli znači da diskriminator misli da je slika lažna, dok vrednost bliža jedinici znači da misli da je slika prava.

3.4 Stage 2

Osnovni cilj druge faze je generisanje slika veće rezolucije na osnovu rezultata dobijenih iz prve faze.

Generator i diskriminatore u ovoj fazi funkcionišu na sličan način kao u prvoj fazi, s tim da postoji ključna razlika u radu generatora. U drugoj fazi, generator koristi informacije iz prve faze. Ovaj proces se odvija na sledeći način: kada se instancira generator u drugoj fazi, potrebno je proslediti mu instancu generatora iz prve faze. Tokom generisanja slika, generator najpre koristi generator dobijen iz prve faze, kako bi dobio odgovarajuću sliku iz prve faze. Potom kodira tu sliku i kombinuje informacije iz kodirane slike sa ostalim informacijama o uslovima, čime generiše novu sliku u drugoj fazi.

Diskriminatore u drugoj fazi procenjuje kako generisane slike visoke rezolucije odgovaraju stvarnim slikama i pruža povratne informacije o njihovom kvalitetu, slično kao i u prvoj fazi.

3.4.1 Greška generatora

Greška generatora (generator loss) meri koliko uspešno generator može da prevari diskriminatore. Cilj generatora je da proizvodi slike koje izgledaju dovoljno stvarno da diskriminatore ne može lako da ih razlikuje od pravih.

Što slike izgledaju uverljivije, to će diskriminatore teže prepoznati da su lažne, što dovodi do smanjenja greške generatora.

Pri računanju grešaka, koristi se Binary Cross-Entropy Loss(BCE) kriterijum. BCE loss kriterijum se koristi za procenu razlike između stvarnih i predviđenih klasa u binarnim klasifikacionim zadacima. Iz tog razloga, pogodan je jer se primenjuje na rezultate diskriminatora koji predstavljaju verovatnoće između 0 i 1, poređenjem sa odgovarajućim labelama klasa.

Greška generatora se izračunava pomoću greške diskriminatora koja se računa za izlaz diskriminatora za lažne slike, poredeći ih sa oznakama pravih slika. Izlaz diskriminatora su vrednosti između 0 i 1 za svaku sliku u batch-u, koje predstavljaju verovatnoću da je slika prava. Vrednost bliža 1 znači da diskriminator smatra da je slika stvarna. Ukoliko generator dobro generiše slike, diskriminator će za te slike davati verovatnoće blizu 1, da su te slike stvarne. Na taj način, izračunata greška za diskriminator će biti manja, prema čemu to doprinosi manjoj grešci generatora.

Osim ove greške, ukupnom gubitku generatora dodaje se i KL divergencija kao regularizator, koja pomaže da distribucija latentnog prostora ostane stabilna, poboljšavajući kvalitet generisanih slika.

```
def generator_loss(netD, fake_imgs, real_labels, conditions, logvar):
    criterion = nn.BCELoss()
    cond = conditions.detach()
    fake = fake_imgs

    fake_features = netD(fake, cond)
    errD_fake = criterion(fake_features.view(-1), real_labels.view(-1))

    kl_loss = KL_loss(conditions, logvar)

    errG_total = errD_fake + kl_loss*2
    return errG_total
```

Figure 5: Generator Loss

3.4.2 Greška diskriminatora

Kao što je već objašnjeno, koristi se BCE Loss kriterijum za računanje greške. Greška diskriminatora se računa na osnovu grešaka za tri vrste parova: stvarne parove, pogrešne parove, i lažne parove. Stvarni parovi se sastoje od pravih slika sa odgovarajućim uslovima, pogrešni parovi uključuju prave slike uparene sa slučajno izabranim pogrešnim uslovima, dok lažni parovi sadrže lažne (generisane) slike sa odgovarajućim uslovima. Ukupna greška diskriminatora je kombinacija ovih vrednosti, pri-

čemu se greške za pogrešne i lažne parove ponderišu da bi se dobila uravnotežena ocena. Greške za stvarne parove dominiraju ukupnim gubitkom kako bi osigurale da diskriminator pouzdano prepoznae stvarne slike i održava stabilnost treniranja.

```
def discriminator_loss(netD, real_imgs, fake_imgs, real_labels, fake_labels, conditions):
    criterion = nn.BCELoss()
    batch_size = real_imgs.size(0)

    cond = conditions.detach()
    fake = fake_imgs.detach()

    # Real parovi
    real_features = netD(real_imgs, cond)

    errD_real = criterion(real_features.view(-1), real_labels.view(-1))

    # Pogrešni parovi (realne slike uparene sa pogrešnim random uslovima)
    wrong_cond = cond[torch.randperm(batch_size)] # Slučajna permutacija
    wrong_features = netD(real_imgs, wrong_cond)
    errD_wrong = criterion(wrong_features.view(-1), fake_labels.view(-1))

    # Lažni parovi
    fake_features = netD(fake, cond)
    errD_fake = criterion(fake_features.view(-1), fake_labels.view(-1))

    errD = errD_real + (errD_fake + errD_wrong)*0.5
    return errD
```

Figure 6: Discriminator Loss

3.5 Trening

Trening prve i druge faze u GAN modelu odvija se kroz sličan postupak, s tim da se razlikuju u nivou detalja i rezoluciji generisanih slika. Svaki trening prolazi kroz unapred definisan broj epoha, pri čemu se u svakoj epohi obrađuje skup batch-eva podataka. U okviru svakog batch-a, generator generiše slike na osnovu datih podataka, nastojeći da one budu što sličnije stvarnim slikama iz skupa podataka.

Diskriminatator potom evaluira te generisane slike, upoređujući ih sa stvarnim slikama iz batch-a i pokušava da ih razlikuje. Na osnovu ovog poređenja izračunavaju se greške za oba modela: greška generatora, koja pokazuje koliko su generisane slike blizu stvarnim, i greška diskriminatatora, koja reflektuje njegovu sposobnost da razlikuje stvarne slike od lažnih.

Na kraju svakog prolaska kroz batch, težine generatora i diskriminatatora se ažuriraju koristeći Adam optimizator. U ovom slučaju, korišćeni su parametri betas=(0.5, 0.999). Parametar beta1=0.5 omogućava bržu reakciju na promene u gradijentima i dinamičkiji proces treniranja, dok parametar beta2=0.999 ima podrazumevanu vrednost.

Ovaj proces omogućava da generator postepeno nauči da proizvodi sve realističnije slike, dok diskriminator postaje sve bolji u identifikovanju lažnih slika.

3.5.1 Trening nad jednom vrstom ptica

Tokom treninga, pokušala sam sa eksperimentom da preprilagodom model za jednu vrstu ptica. Zaključila sam da se bolji rezultati dobijaju kada povećam learning rate. Konkretno, inicijalno learning rate je bio 0.0002 i za generator i za diskriminator. Ali pri treniranju prve faze nad jednom vrstom, model nije uspevao da nauči korisne karakteristike, pa sa povećanjem learning rate-a generatora na 0.0005 (i više) uspevam da dobijem sledeće rezultate:



Figure 7: Realne slike

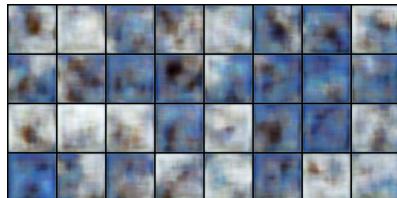


Figure 8: Stage 1

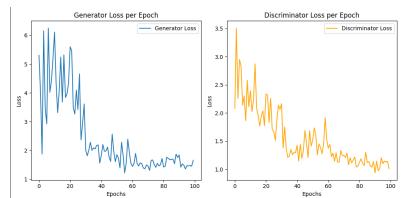


Figure 9: Greške u prvoj fazi

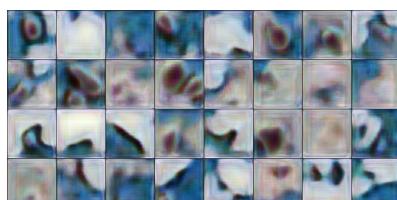


Figure 10: Stage 2

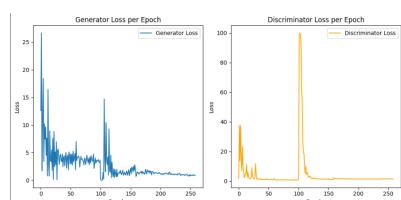


Figure 11: Greške u drugoj fazi

Zaključujemo da slike u drugoj fazi jesu malo preciznije nego slike u prvoj fazi, kao i to da su boje uglavnom naučene i pogodjene. Takodje,

uspela sam da podesim parametre tako da se i greška generatora i greška diskriminatora smanjuju, barem za prvu fazu.

3.5.2 Trening nad 40 vrsti ptica

Trening nad 40 vrsti ptica je bio sličan kao i trening nad manjim brojem ptica.

Za veoma mali learning rate generatora (ispod 0.0001), dobijeni su apstraktniji rezultati za 20 odnosno 40 vrsti ptica. Nažalost, za ove rezultate nemam podatke o greškama.



Figure 12: stage 1, 20 vrsti



Figure 13: stage 1, 40 species

Nakon ovog inicijalnog treniranja, dobijaju se drugačiji rezultati kada se radi sa većim learning rate-om. Možemo videti rezultate dva uzastopna treniranja pri čemu je samo povećana vrednost learning rate-a za generator sa 0.0002 na 0.0005.



Figure 14: Realne slike, 40 vrsti



Figure 15: stage 1, lr=0.0005

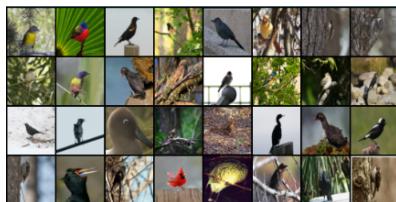


Figure 16: Realne slike, 40 vrsti



Figure 17: stage 1, lr=0.0002

Zaključujemo da za veći learning rate, generator generiše nešto preciznije slike, sa jasnijim oblicima. Nažalost, u oba slučaja, dobija se

situacija gde diskriminatot nadjačava generator, a to možemo videti na sledećoj slici koja je dobijena treningom :

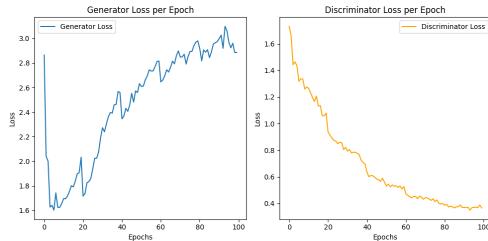


Figure 18: Greške, 40 vrsta

Sledi rezultat još jednog treninga nad 40 vrsti ptica, pri čemu je korišćena drugaćija transformacija ulaznih slika (vidimo da su mnoge slike odsečene i drastično promenjene), pri čemu je learning rate za generator značajno povećan (iznad 0.0007).



Figure 19: Realne slike



Figure 20: Stage 1

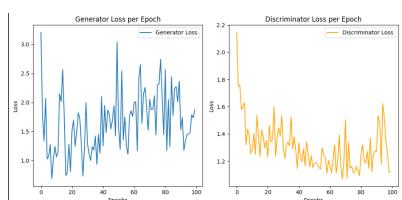


Figure 21: Greške u prvoj fazi

Što se tiče treninga druge faze za 40 vrsta ptica, nažalost, zbog tehničkih ograničenja nisam uspela da ga završim do kraja.

4 Test

Što se tiče faze testiranja, ona je implementirana i radi nad nezavisnim skupom test podataka. Nažalost, rezultati koje dobijamo generisanim

modelom su neodgovarajući. Očigledno da je model preprilagođen, i nije zaista naučio da prepoznaće odgovarajuće karakteristike ptica.

Rezultati testiranja nad skupom 40 vrsti ptica, primenom dobijenog modela generatora u prvoj fazi:

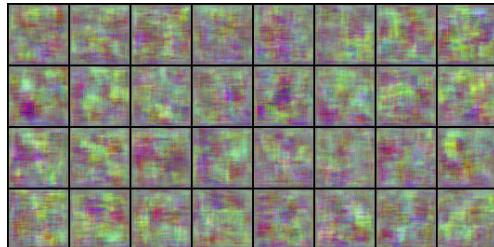


Figure 22: Test 1

5 Rezultati i zaključak

Tokom treninga, nisam uspela da dođem do zadovoljavajućih rezultata, odnosno da generišem slike ptica na osnovu zadatih tekstualnih opisa. Slike koje sam dobila na kraju treninga prve faze, nalikuju izvornim slikama ptica po bojama i oblicima koji se naziru. Takođe, zbog tehničkih problema i kompleksne arhitekture modela, nisam uspela da dobijem rezultate druge faze za veći broj vrsti odnosno slika. Na osnovu druge faze nad jednom vrstom, uspela sam da zaključim da druga faza uspešno prepoznaće osnovne elemente na slikama prve faze, i dodatno unapređuje slike iz prve faze.

U određenim slučajevima, model uspeva da nauči neke karakteristike dok u nekim drugim ne. Na primer, pri treniranju, za opis 'a small bird with gray head, brown wings and gray and brown belly' dobijam sledeću sliku:



Figure 23: small bird with gray head

Tokom treninga, suočavala sam se sa brojnim problemima, kao što su neuravnoteženost obuke generatora i diskriminatora, nedovoljna količina

podataka, Mode Collapse...

Na početku, obuku sam počela sa svih 200 vrsti ptica, gde sam ubrzo shvatila da će trening biti previše vremenski iscrpan i potom smanjila ukupan broj vrsti na svega 10. Nakon nekog vremena, došla sam do zaključka da model potencijalno ne uspeva da nauči karakteristike ptica jer nema dovoljno podataka, pa sam postepeno povećavala broj vrsti, prvo na 20, pa na 40. Usput sam pokušala i da preprilagodim model za samo jednu vrstu gde sam zaključila da model ispravno uči boje i slike nad jednom vrstom, kao i da veći learning rate u ovom slučaju značajno utiče na bolju obuku, ali i koliko je bitna dobra inicijalizacija težina. Takođe sam eksperimentisala i pratila kako se model ponaša za različita izračunavanja grešaka.

Imala sam i problem sa pojmom koja se zove Mode Collapse. To se dešava kada se generator počne da proizvodi samo jednu ili nekoliko sličnih slika, jer taj specifičan izlaz generatora minimizuje grešku u odnosu na diskriminator. Zbog toga, umesto da generiše širok spektar različitih slika, generator stalno pravi slične ili iste slike, pa na kraju treninga sve generisane slike postaju identične.

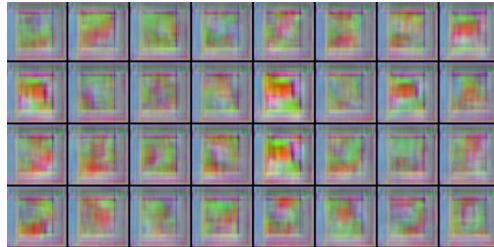


Figure 24: Mode collapse

Pored svega toga, uočila sam da se često javljaju problemi kada jedan deo modela, uglavnom diskriminator, postane previše precizan dok drugi ne uspeva da postigne sličan nivo tačnosti. Ovaj problem je bio najizazovniji, i pokušavala sam da ga rešim na više načina, promenom ključnih parametara kao što su learning rate (i generator ai diskriminatora), GF_dim i DF_dim, z_dim, ali i različitim optimizatorima ... Podešavanjem ovih parametara uspela sam da postignem nešto bolju ravnotežu između generatora i diskriminatora, čime se poboljšala ukupna stabilnost i performanse modela.

Iako nisam došla do očekivanih rezultata, došla sam do mnogobrojnih zaključaka prilikom rada na GAN mrežama. Glavni zaključci koje sam izvela:

- Veći learning rate generatora nam daje slike jasnijih i definisanijih oblika i boja, dok nam manji daje apstraktnije slike, nalik umetnosti
 - Manji learning rate diskriminadora nam omogućuje da diskriminator sporije uči, kako bismo uspeli da izbegnemo situaciju u kojoj diskriminator nadjačava generator
 - Veća vrednost parametra GF_DIM koji predstavlja osnovni broj filtera u prvom sloju generatora, nam daje slike sa više detalja i tekstura. Što je njegova vrednost veća, to model dobija veću sposobnost da uči složenije karakteristike. Prevelika vrednost ovog parametra može dovesti do povećane složnosti modela i problema sa treniranjem.
 - Veća vrednost parametra DF_DIM koji predstavlja predstavlja osnovni broj filtera u prvom sloju diskriminatora, pomaže diskriminatoru da preciznije razlikuje prave od lažnih slika.
 - Veća vrednost parametra Z_DIM, koji predstavlja dimenzionalnost latentnog vektora (šuma), omogućava generatoru da koristi širi i kompleksniji latentni prostor, što povećava raznovrsnost i detaljnost generisanih slika, omogućavajući mu da obuhvati više varijacija u karakteristikama, ali istovremeno povećava računske zahteve i složenost treniranja zbog obrade većeg broja dimenzija.
 - Inicijalizacija težina mreža značajno utiče na generisane slike jer pravilna inicijalizacija može ubrzati konvergenciju, izbeći lokalne minimume i stabilizovati proces treniranja, dok loša inicijalizacija može rezultirati lošim kvalitetom slika i nedostatkom raznovrsnosti.
- Primer:

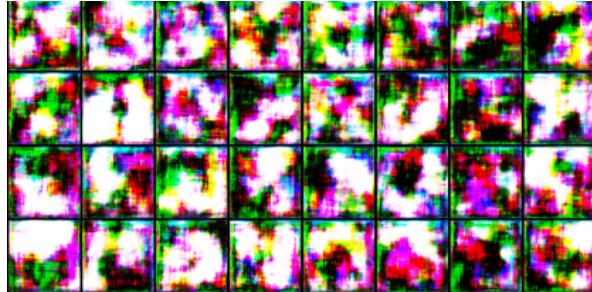


Figure 25: Loša inicijalizacija

- Različite transformacije ulaznih slika doprinose većoj raznovrsnosti i poboljšavaju učenje modela. Ipak, previše drastične transformacije mogu značajno izmeniti ulazne slike (npr. velike razlike u boji,

sečenje i slične promene) do te mere da ni čovek, a ni model, ne mogu prepoznati da se na slici nalazi ptica.

- Korišćenje SGD optimizatora za diskriminatore umesto Adam-a može dovesti do nestabilnog treninga, pri čemu sve generisane slike počinju da liče jedna na drugu, što rezultira mode collapse pojavom. Ideja za korišćenje SGD-a se zasniva na tome da SGD usporava konvergenciju diskriminatora, omogućavajući generatoru više vremena da se prilagodi i postigne bolju ravnotežu između ova dva modela. Međutim, u praksi se pokazalo da ovakav izbor optimizatora doprinosi još većoj nestabilnosti između generatora i diskriminatara.

Tokom ovog rada, zaključila sam da rad na GAN modelima predstavlja izuzetno kompleksnu temu koja zahteva temeljno razumevanje i pažljivo podešavanje parametara. Obuka GAN-ova je dosta izazovna kada je u pitanju uspostavljanje balansiranog treninga zmeđu generatora i diskriminatora, i definitivno je tema na kojoj uvek može da se dodatno radi i unapređuje.

References

- [1] <https://github.com/zeusm9/Text-to-Photo-realistic-Image-Synthesis-with-Stacked-Generative-Adversarial-Networks/tree/master>
- [2] <https://github.com/savya08/StackGAN/tree/master>
- [3] Cristian Bodnar, *Text to Image Synthesis Using Generative Adversarial Networks*. University of Manchester, 2018.
- [4] Han Zhang, Tao Xu, Hongsheng Li3, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas *StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks*. Rutgers University, Lehigh University, The Chinese University of Hong Kong, Baidu Research