

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINARSKI RAD

GENERIRANJE SLIKA IZ TEKSTA

Matej Magat

Voditelj: izv. prof. dr. sc. Goran Delač

Zagreb, lipanj, 2025.

Generiranje slika iz teksta

Matej Magat

Sažetak

Ovaj seminarski rad istražuje i uspoređuje različite pristupe generiranju slika iz teksta, uključujući generativne suparničke mreže (GAN-ove), difuzijske modele, autoregresivne modele temeljene na transformatorima i velike jezične modele (LLM-ove). Osim toga, rad predstavlja implementaciju i evaluaciju GAN sustava posebno osmišljenog za generiranje slika iz teksta.

Ključne riječi: Duboko učenje; Generativna umjetna inteligencija; Sinteza slika; Generiranje slika iz teksta

Sadržaj

Sažetak	1
1. Uvod	4
2. pristupi generiranju slika iz teksta	5
2.1. Generativne suparničke mreže	5
2.1.1. Arhitektura DCGAN-a	5
2.1.2. Način učenja	6
2.2. Difuzijski modeli	6
2.2.1. Komponente latentnog DM-a	7
2.2.2. Način učenja	7
2.3. Transformeri	8
2.3.1. Arhitektura transformera	8
2.3.2. Mehanizam samopozornosti	9
2.3.3. Transformatorski modeli za generiranje slika	9
2.3.4. LLM-ovi i generiranje slike iz teksta	10
3. Implementirani GAN sustav	11
3.1. Cilj implementiranog modela i skup podataka	11
3.2. Detalji implementacije	11
3.2.1. Odabir okvira za razvoj	11
3.2.2. Dizajn arhitekture DCGAN-a	12
3.2.3. Inicijalizacija težina	13
3.3. Konfiguracija treniranja i pregled gubitaka	14
3.3.1. Upravljanje stope učenja	16
3.3.2. Prikaz gubitaka tijekom treniranja	17

3.4. Odabir najboljeg modela	18
3.4.1. Prikaz greške na skupu za testiranje	18
4. Rezultati	20
5. Zaključak	22
Literatura	23

1. Uvod

Područje generiranja slika na temelju teksta usmjereno je na razvoj sustava koji mogu generirati realistične slike na temelju tekstualnih opisa. Primjene ove tehnologije obuhvaćaju generiranje kreativnog sadržaja i digitalne umjetnosti, pomoć osobama s oštećenjem vida te unaprjeđenje interakcije čovjeka i računala. Budući da je odnos između jezika i vizualnog sadržaja složen i višeznačan, generiranje visokokvalitetnih slika iz teksta zahtijeva napredne metode strojnog i dubokog učenja. Među najistaknutijim pristupima nalaze se generativne suparničke mreže (generative adversarial networks, GAN-ovi), difuzijski modeli, autoregresivni modeli temeljeni na transformatorima te, u novije vrijeme, veliki jezični modeli koji povezuju tekstualne i slikovne modalitete. Ovaj rad daje pregled navedenih pristupa, ističući njihove temeljne principe i komparativne prednosti. Poseban naglasak stavljen je na implementaciju i evaluaciju sustava temeljenog na GAN-u, prilagođenog za generiranje slika iz teksta, čime se ilustriraju praktični izazovi i mogućnosti u ovom brzo razvijajućem području.

2. pristupi generiranju slika iz teksta

2.1. Generativne suparničke mreže

Generativne suparničke mreže (GAN-ovi) su duboke neuronske mreže koje se sastoje od dvije komponente:

- **Generator** – stvara sintetičke podatke (npr. slike)
- **Diskriminator** – razlikuje stvarne od generiranih uzoraka

Obje komponente uče istovremeno kroz suparničko natjecanje. Generator stvara sintetičke podatke, a diskriminator pokušava odrediti je li uzorak stvaran ili generiran. Učenje se zaustavlja kada diskriminator više ne može raspoznati je li uzorak stvaran ili generiran.

Ovu minimax igru opisuje funkcija:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.1)$$

2.1.1. Arhitektura DCGAN-a

Dubokokonvolucijski GAN (DCGAN) [1] unaprijeđuje osnovnu arhitekturu kroz:

- Zamjenu slojeva sažimanja **proširenim konvolucijama**
- Primjenu **normalizacije po grupama** u oba modela
- **ReLU** aktivacije u generatoru (izuzev izlaznog sloja s tanh)
- **LeakyReLU** u diskriminatoru ($\alpha = 0.2$)

2.1.2. Način učenja

1. **Rekurentna neuronska mreža** (RNN) kodira opise prirodnog jezika u vektorske reprezentacije.
2. **Generator** kombinira vektor šuma s umetnutim tekstom kako bi sintetizirao slike koje su usklađene s tekstualnim unosom.
3. **Diskriminator** ocjenjuje i realističnost generiranih slika i njihovu semantičku dosljednost s tekstom.

2.2. Difuzijski modeli

Difuzijski modeli su generativni sustavi koji uče distribuciju podataka kroz iterativno dodavanje i uklanjanje šuma. [2] Proces uključuje:

1. **Unaprijedna difuzija** koja postepeno dodaje šum na ulazne podatke x_0 kroz T koraka:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t \mathbf{I}) \quad (2.2)$$

Gdje β_t definira raspored šuma.

2. **Difuzija u suprotnom smjeru** u kojoj neuronska mreža ϵ_θ uči uklanjati šum minimizacijom funkcije gubitka:

$$\mathcal{L}_{\text{DM}} = \mathbb{E}_{x, \epsilon, t} \left[\|\epsilon - \epsilon_\theta(x, t)\|_2^2 \right] \quad (2.3)$$

Tradicionalni DM-ovi u piksel prostoru suočavaju se s računalnim ograničenjima pri generiranju slika visoke rezolucije ($>512\text{px}$), zahtijevajući oko 1,5 GB memorije po uzorku i više od 1000 koraka uklanjanja šuma. Prebacujući difuzijski proces u **smanjeni latentni prostor**, rješava se problem skalabilnosti difuzijskih modela. [3]

2.2.1. Komponente latentnog DM-a

1. **KL-regularizirani autoenkoder** \mathcal{E}/\mathcal{D} komprimira slike u latentne kodove:

$$z = \mathcal{E}(x)$$

(npr. 64×64 za ulaze od 512px) Održava perceptivnu vjernost uz 48× kompresiju prostora.

2. **Difuzija u latentnom prostoru** s pomoću U-Net:

- ResNet blokove za hijerarhijsko izdvajanje značajki
- Cross-attention slojeve za multimodalno kondicioniranje
- Self-attention za globalni kontekst

3. **Mehanizmi kondicioniranja**

- Tekstualni promptovi kodirani putem CLIP ViT-L/14
- Cross-attention mapira tekstualne ugradnje $\tau_\theta(y)$ na latentne značajke:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

gdje je $Q = W_Q \cdot \varphi(z_t)$, a $K/V = W_K \cdot \tau_\theta(y)$

2.2.2. Način učenja

Optimizacija u dva koraka [4]:

- **Korak 1:** Treniranje autoenkodera s:
 - Perceptivnim gubicima (L1 + SSIM)
 - Adversarijalnim gubicima (GAN objektiv)
- **Korak 2:** Zamrzavanje enkodera i treniranje difuzije:
 - 200k koraka optimizacije na LAION-400M skupu

- Učenje u latentnom prostoru dimenzija $64 \times 64 \times 4$

2.3. Transformeri

Transformeri su napredna vrsta neuronskih mreža koja je revolucionirala obradu podataka poput teksta i slika. Za razliku od starijih modela poput RNN-ova (koji obrađuju podatke slijedno) ili CNN-ova (koji se fokusiraju na lokalne obrasce), transformeri rade paralelno i analiziraju cijeli kontekst odjednom. [5]

Svojstva transformera [6]:

1. **Mehanizam samopozornosti** – automatski identificira koje dijelove ulaza (npr. riječi u rečenici ili dijelove slike) treba povezati, čak i ako su udaljeni
2. **Paralelna obrada** – svi elementi ulaza se analiziraju istovremeno, što ubrzava treniranje i omogućuje rad s dugim nizovima podataka

2.3.1. Arhitektura transformera

Transformeri koriste dvokomponentni dizajn [7]:

Enkoder

Obrada ulaznih sekvenci (tekst, slike itd.) u kontekstualizirane reprezentacije kroz:

- **Ulazni ugradbeni vektori:** Vektorske reprezentacije tokena (riječi/segmenti slika)
- **Pozicijsko kodiranje:** Sinusne/kosinusne funkcije za opis prostornih/vremenskih odnosa
- **Multi-head self-attention:** Paralelne pozornostne "glave" modeliraju različite odnose
- **Unaprijedne mreže:** Nelinearne transformacije za profinjenje reprezentacija

Dekoder

Generiranje izlaznih sekvenci autoregresivnim pristupom:

- **Maskirana samopozornost:** Sprječava "curenje" informacija o budućim tokenima
- **Unakrsna pozornost:** Usmjerava dekodera na relevantne dijelove enkodirovog izlaza
- **Izlazna projekcija:** Softmax sloj za predviđanje tokena/piksela

2.3.2. Mehanizam samopozornosti

Ključna inovacija koja omogućuje globalno modeliranje konteksta:

Komponenta	Funkcionalnost
Query/Key/Value vektori	Projekcija embeddinga u prostor relacija
Alokacijski rezultati	Skalirani umnošci određuju težine pozornosti
Multi-head obrada	Paralelne "glave" hvataju različite obrasce

Ova arhitektura omogućuje paralelnu obradu svih tokena uz $O(1)$ put između bilo koja dva tokena, za razliku od $O(n)$ sekvencijalne ovisnosti u RNN-ovima. [8]

2.3.3. Transformatorski modeli za generiranje slike

Modeli za generiranje slike iz teksta temeljeni na arhitekturi transformatora obavljaju obradu ulaznih podataka kroz tri faze[9]:

1. Kodiranje teksta

Enkoder obrađuje ulazni tekst (npr. "crvena mačka koja vozi bicikl") koristeći tokenizaciju i mehanizme samopozornosti za stvaranje kontekstualiziranih odnosa između riječi. Ovo proizvodi guste vektorske reprezentacije (embeddings) koji bilježe semantičko značenje.

2. Predviđanje tokenskih slika

Dekoder koristi slojeve unakrsne pozornosti za poravnavanje tekstualnih embeddinga s vizualnim konceptima, autoregresivno predviđajući diskretne tokene slike. Svaki korak generiranja tokena fokusira se na specifične aspekte teksta kroz težine pozornosti - na primjer, naglašava "crvena" pri predviđanju tokena boje.

3. Rekonstrukcija slike

Predviđeni tokeni se pretvaraju u piksele putem odvojenih modela dekodiranja kao što su VQ-VAE dekoderi. Ova završna faza preslikava diskretni slijed tokena u kontinuirani prostor slike uz očuvanje semantičkog poravnanja s izvornim tekstom.

Komponenta	Funkcija	Ključni mehanizam
Enkoder	Razumijevanje teksta	Samopozornost
Dekoder	Poravnavanje vizualnih koncepata	Unakrsna pozornost
Rekonstrukcija	Pretvorba tokena u sliku	Vektorska kvantizacija

Primjer: DALL-E

[9] [10] Prilikom obrade uputa "crvena mačka koja vozi bicikl", DALL-E-ov dekodek sekvencijalno predviđa:

- Tokene oblika koji ocrtavaju morfologiju mačke
- Tokene boje koji specificiraju RGB vrijednosti za dlaku
- Prostorne tokenne koji pozicioniraju komponente bicikla

VQ-GAN dekodek zatim pretvara ove tokenne u sliku od 256x256 piksela.

2.3.4. LLM-ovi i generiranje slike iz teksta

LLM model interpretira strukturu, semantiku i kontekst korisničkog unosa, razlažući ga na značajne komponente kao što su objekti, atributi, odnosi i kontekst scene. LLM zatim kodira to razumijevanje u visokodimenzionalnu vektorsku reprezentaciju koja služi kao semantički nacrt za kasniji proces generiranja slike. [11]

Sama generacija slike obično se provodi pomoću generativnih modela kao što su difuzijski modeli ili generativne suparničke mreže.

3. Implementirani GAN sustav

Repozitorij implementiranog sustava [12]

3.1. Cilj implementiranog modela i skup podataka

Cilj ovog modela je stvaranje realističnih slika ptica iz tekstualnih opisa, koristeći DC-GAN arhitekturu za sintezu slike iz teksta. Ovaj zadatak zahtijeva modele koji mogu precizno interpretirati fino detaljirane tekstualne opise i proizvesti odgovarajuće detaljne slike. Za pretvaranje tekstualnih opisa u vektorske reprezentacije korišten je BERT enkoder, koji omogućuje efikasno sažimanje semantičkih značajki prirodnog jezika. Ovi vektori zatim služe kao ulaz DCGAN modelu za generiranje slika.

Model je učen na skupu podataka Caltech-UCSD Birds 200-2011 (CUB-200-2011), koji sadrži 11.788 slika ptica iz 200 vrsta. Svaka slika popraćena je s više prirodnih jezičnih opisa, kao i s oznakama kao što su okviri, lokacije dijelova i binarni atributi. U početku, podaci su podijeljeni na 80% za učenje i 20% za testiranje.

Također je implementiran postupak učenja koji koristi validacijski skup i upravljač stope učenja. Za ovu fazu, podaci su podijeljeni na 80% za učenje, 10% za validaciju i 10% za testiranje.

3.2. Detalji implementacije

3.2.1. Odabir okvira za razvoj

Model je implementiran u Pythonu koristeći PyTorch 2.7.0[13], iskorištavajući njegov sustav automatske diferencijacije i operacije na tenzorima ubrzane na grafičkoj kartici. Treniranje je provedeno koristeći CUDA 11.8 na grafičkoj kartici NVIDIA GeForce RTX

2050, što omogućuje učinkovitu obradu slika veličine 64x64 u serijama.

3.2.2. Dizajn arhitekture DCGAN-a

Arhitektura slijedi načela DCGAN-a s pažljivim odabirom komponenti.[14]

Arhitektura diskriminatora

Listing 3.1: Pytorch implementacija Diskriminatora

```
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.main = nn.Sequential(
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )
```

Obrazloženje dizajna:

- **Konvolucije s korakom:** Koristi jezgre 4x4 s korakom 2 za postepeno smanjivanje prostornih dimenzija
- **LeakyReLU aktivacija ($\alpha = 0.2$):** Omogućuje protok gradijenta za negativne ulaze
- **Selektivni BatchNorm:** Primjenjuje se nakon početnih slojeva
- **Dizajn uskog grla:** Konačna 4x4 konvolucija smanjuje dimenzije na 1x1

Arhitektura generatora

Listing 3.2: Pytorch implementacija Generatora

```
class Generator(nn.Module):
```

```

def __init__(self):
    super(Generator, self).__init__()
    self.main = nn.Sequential(
        nn.ConvTranspose2d( nz, ngf * 8, 4, 1, 0, bias=False),
        nn.BatchNorm2d(ngf * 8),
        nn.ReLU(True),
        nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
        nn.BatchNorm2d(ngf * 4),
        nn.ReLU(True),
        nn.ConvTranspose2d( ngf * 4, ngf * 2, 4, 2, 1, bias=False),
        nn.BatchNorm2d(ngf * 2),
        nn.ReLU(True),
        nn.ConvTranspose2d( ngf * 2, ngf, 4, 2, 1, bias=False),
        nn.BatchNorm2d(ngf),
        nn.ReLU(True),
        nn.ConvTranspose2d( ngf, nc, 4, 2, 1, bias=False),
        nn.Tanh()
    )

```

Obrazloženje dizajna:

- **Transponirane konvolucije:** Učinkovito uvećava prostorne dimenzije
- **ReLU aktivacija:** Omogućuje zasićene izlaze boja
- **BatchNorm:** Primjenjuje se na svim slojevima osim izlaznog
- **Tanh izlaz:** Ograničava vrijednosti piksela na [-1,1]

3.2.3. Inicijalizacija težina

Model koristi prilagođenu inicijalizaciju težina:

Listing 3..3: Funkcija za inicijalizaciju težina modela

```

def weights_init(m):
    classname = m.__class__.__name__
    if classname.find('Conv') != -1:
        if hasattr(m, 'weight') and m.weight is not None:
            nn.init.normal_(m.weight.data, 0.0, 0.02)
    elif classname.find('BatchNorm') != -1:
        if hasattr(m, 'weight') and m.weight is not None:
            nn.init.normal_(m.weight.data, 1.0, 0.02)
        if hasattr(m, 'bias') and m.bias is not None:
            nn.init.constant_(m.bias.data, 0)
    elif classname.find('Linear') != -1:
        if hasattr(m, 'weight') and m.weight is not None:

```

```

nn.init.normal_(m.weight.data, 0.0, 0.02)
if hasattr(m, 'bias') and m.bias is not None:
    nn.init.constant_(m.bias.data, 0)

```

Tablica 3.1. Strategija inicijalizacije težina

Komponenta	Inicijalizacija
Konvolucijski sloj	$\mu = 0.0, \sigma = 0.02$
BatchNorm težine	$\mu = 1.0, \sigma = 0.02$
BatchNorm pomaci	konstanta 0

Ova implementacija kombinira najbolje prakse DCGAN-a s mogućnošću modernih alata, osiguravajući stabilno adversarijalno treniranje.

3.3. Konfiguracija treniranja i pregled gubitaka

DCGAN model je treniran na CUB-200-2011 skupu podataka s podjelom na 80% podataka za treniranje i 20% za testiranje. Treniran je kroz 750 epoha s stopom učenja od 0.0002. Primarna funkcija gubitka za diskriminator i generator je Binarna unakrsna entropija (BCE), odabrana jer odgovara binarnoj klasifikacijskoj zadaći razlikovanja stvarnih i lažnih slika, kao u originalnom DCGAN radu[15]. Dodatno su korišteni L1 i L2 gubici s koeficijentima 50 i 100. L1 gubitak potiče generator da stvara slike slične stvarnim na razini piksela, dok L2 gubitak usklađuje statističke značajke generiranih i stvarnih slika kroz usporedbu srednjih aktivacija u diskriminatoru.

Tijekom treniranja, diskriminator procjenjuje tri vrste slika:

- **Stvarne slike** označene kao 1,
- **Lažne slike** označene kao 0,
- **Pogrešne stvarne slike** (nasumični uzorci iz skupa podataka koji ne odgovaraju ulaznom embeddingu) također označene kao 0.

Uključivanje pogrešnih slika osnažuje diskriminator da nauči robustnije značajke i sprječava generator da zanemari ulazne embedinge, osiguravajući da generira slike relevantne za zadani opis.

Gubitak generatora kombinira adversarijalni BCE gubitak (namijenjen varanju dis-

kriminatora) s L1 i L2 gubicima za rekonstrukciju i usklađivanje značajki. Ovaj više-objektivni pristup balansira oštrinu i realističnost generiranih slika s vjernošću ulaznim embedinzima. Koeficijenti L1 i L2 postavljeni su tako da prioritiziraju dosljednost značajki i kvalitetu rekonstrukcije, zadržavajući pritom adversarialnu dinamiku ključnu za GAN-ove.

Gubitak diskriminatora izražava koliko dobro diskriminator razlikuje stvarne podatke od onih koje je generirao generator. Diskriminator je u biti binarni klasifikator koji za svaki uzorak procjenjuje je li stvaran ili lažan. Kad je gubitak diskriminatora nizak, to znači da diskriminator uspješno razlikuje stvarne od generiranih podataka. Ako je gubitak diskriminatora visok ili se približava vrijednosti koja ukazuje na nasumično pogađanje (npr. 0.5 za BCE gubitak), to znači da diskriminator ne može pouzdano razlikovati stvarne i lažne uzorke, što može biti znak da generator proizvodi vrlo realistične slike ili da diskriminator još nije dovoljno naučio relevantne značajke.

Listing 3.4: Pytorch treniranje DCGAN-a

```
# Treniranje diskriminatora
discriminator_optim.zero_grad()
noise = torch.randn(batch_size, noise_dim, 1, 1, device=device)
fake_images = generator(noise, embeddings)
real_out, real_act = discriminator(images, embeddings)
d_loss_real = criterion(real_out, torch.full_like(real_out, real_label, device=device)
)
wrong_out, wrong_act = discriminator(wrong_images, embeddings)

d_loss_wrong = criterion(wrong_out, torch.full_like(wrong_out, fake_label, device=
device))
fake_out, fake_act = discriminator(fake_images.detach(), embeddings)
d_loss_fake = criterion(fake_out, torch.full_like(fake_out, fake_label, device=device)
)

d_loss = d_loss_real + d_loss_wrong + d_loss_fake
d_loss.backward()
discriminator_optim.step()

# Treniranje generatora
generator_optim.zero_grad()
noise = torch.randn(batch_size, noise_dim, 1, 1, device=device)
fake_images = generator(noise, embeddings)
out_fake, act_fake = discriminator(fake_images, embeddings)
out_real, act_real = discriminator(images, embeddings)

g_bce = criterion(out_fake, torch.full_like(out_fake, real_label, device=device))
```



```
g_l1 = l1_coef * l1_loss(fake_images, images)
g_l2 = l2_coef * l2_loss(torch.mean(act_fake, 0), torch.mean(act_real, 0).detach())

g_loss = g_bce + g_l1 + g_l2
g_loss.backward()
generator_optim.step()
```

Ovaj pristup dao je vrlo dobre rezultate, koji su prikazani u odjeljku Rezultati 4.

3.3.1. Upravljanje stope učenja

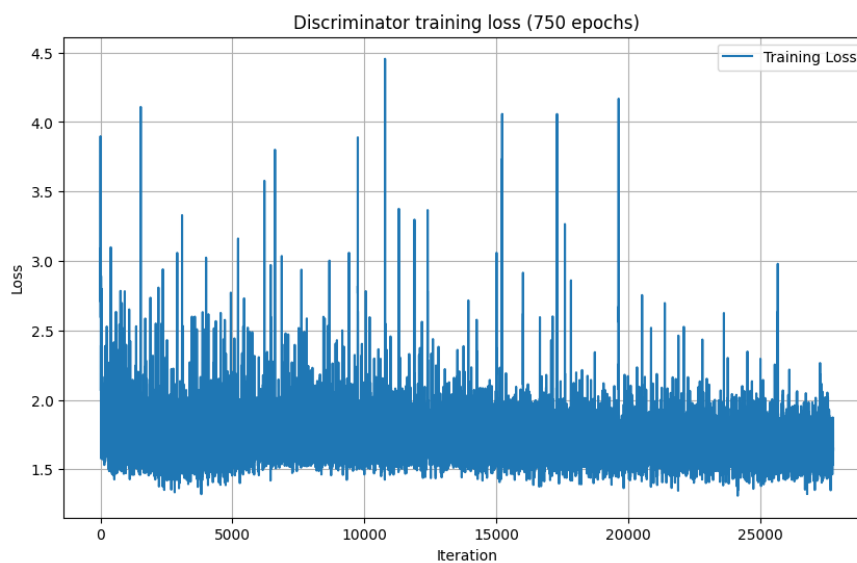
Da bismo procijenili utjecaj upravljača stope učenja na stabilnost treniranja DCGAN-a, implementirane su dvije primarne strategije podjele skupa podataka uz različite konfiguracije upravljača stope učenja.

Prvi pristup dodjeljivao je 75% podataka za obuku, s 15% za validaciju i testiranje, dok je drugi koristio omjer 80:10:10. Obje konfiguracije koristile su ReduceLROnPlateau upravljač stope učenja s ključnim parametrima postavljenim za praćenje gubitka validacije, uključujući period strpljenja od 15 epoha i minimalni prag stope učenja od 1e-6, počevši od početne stope od 0,0002.

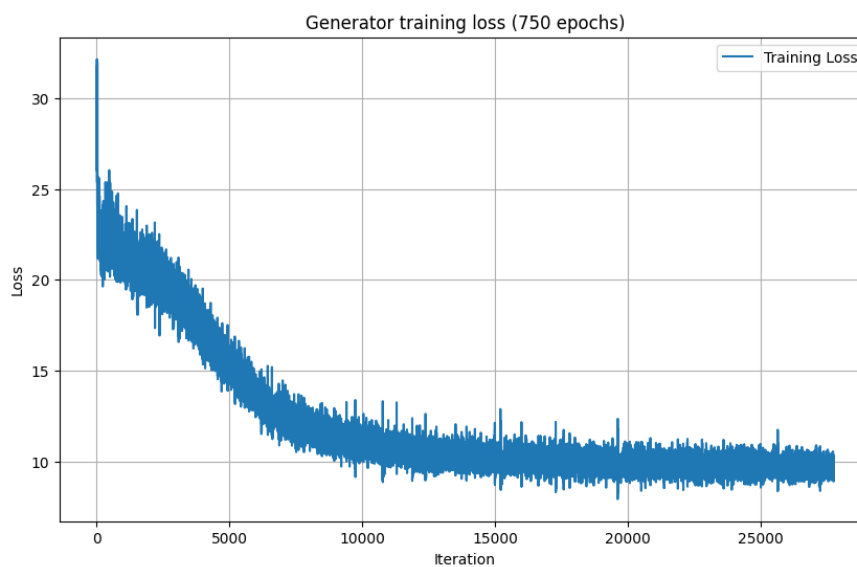
Kao što je već spomenuto, osnovni model koristio je fiksnu stopu učenja bez mehanizama za upravljanje stope učenja. Usporedna ispitivanja uvela su dvije varijante rasporeda: agresivnu konfiguraciju s faktorom smanjenja 0,5 (prepolovljenje stope učenja nakon detekcije platoa) i umjerenu verziju s faktorom 0,95 (postupna smanjenja od 5%). Metrike treniranja otkrile su da su obje implementacije rasporeda uzrokovale preuranjenu konvergenciju u suboptimalna područja gubitka unutar 40-50 epoha, za razliku od kontinuirane optimizacije osnovnog modela tijekom preko 200 epoha.

Raspored stope učenja nije kompatibilan s dinamikom GAN ravnoteže. Konvencionalne metode detekcije platoa često pogrešno tumače prolazna ravnotežna stanja između generatorskih i diskriminatorskih mreža kao prave optimizacijske plateau, što dovodi do preuranjenih prilagodbi stope učenja.[16]

3.3.2. Prikaz gubitaka tijekom treniranja



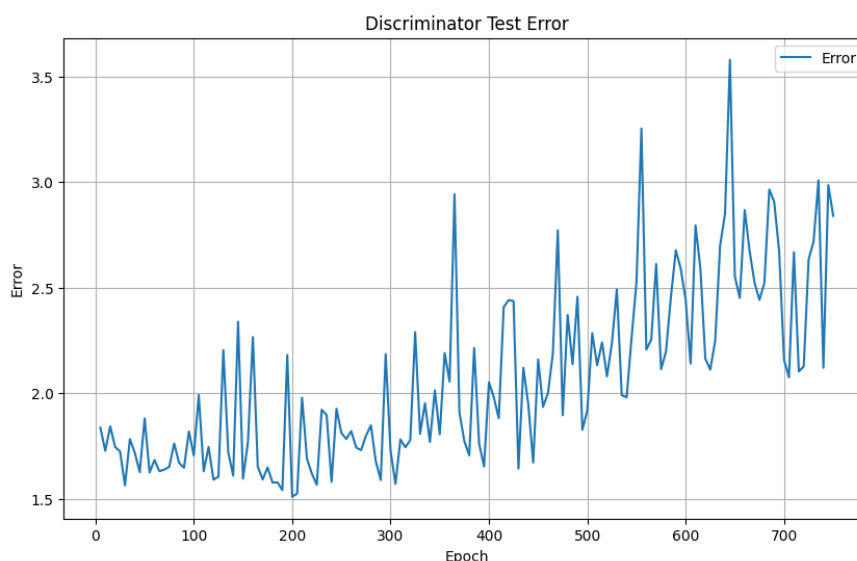
Slika 3.1. Gubitak diskriminatora tijekom treniranja



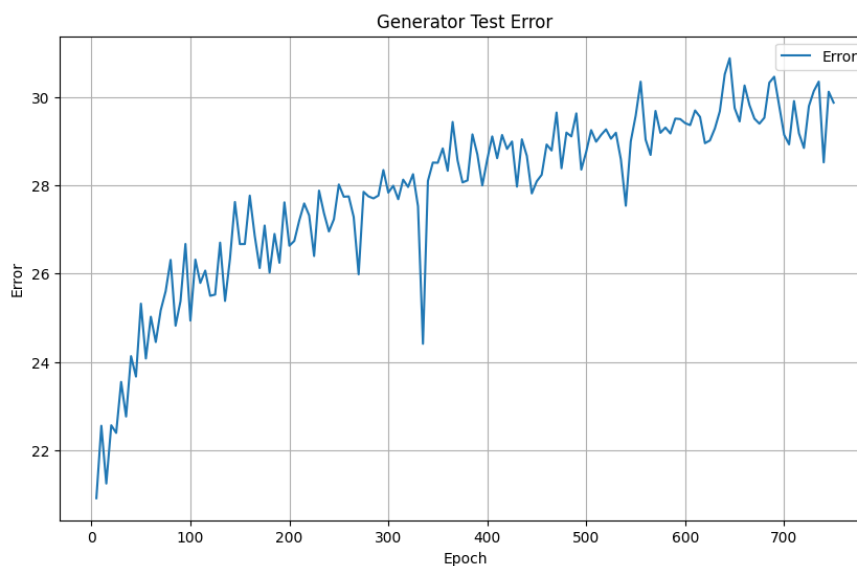
Slika 3.2. Gubitak generatora tijekom treniranja

3.4. Odabir najboljeg modela

3.4.1. Prikaz greške na skupu za testiranje



Slika 3.3. Gubitak diskriminatora tijekom treniranja

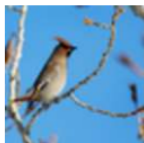
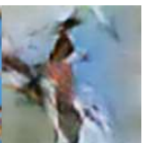
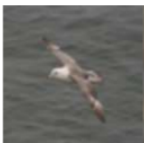
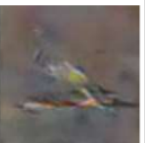


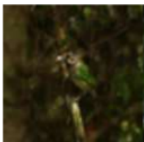





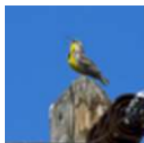
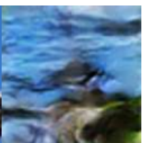




Slika 3.4. Gubitak generatora tijekom treniranja

Optimalni model postiže najnižu vrijednost gubitka (1.51) nakon 200 epoha treniranja, što ukazuje na uspješnu suradnju generatora i diskriminatora. Gubitak diskriminatora različit od nule (1.51) potvrđuje da ne postiže savršenu točnost u razlikovanju stvarnih i generiranih slika, što je ključno za održavanje dinamičke ravnoteže u GAN arhitekturi. Istovremeno, vrijednost daleko od 0.5 isključuje mogućnost nasumičnog pogađanja, što

ukazuje na smislenu adaptaciju modela. U intervalu 200–205 epoha diskriminator pokazuje izuzetnu stabilnost, s fluktuacijama gubitka manjim od 1%, dok generator u epohi 200 bilježi umjerenu grešku od 26.63. Sljedećih pet epoha prati postupni rast generatorova gubitka od 4%, ali bez znakova eksponencijalne divergencije. Ova ravnoteža omogućuje generatoru da proizvodi konzistentne rezultate, dok diskriminator ostaje dovoljno osjetljiv da pruža korisne povratne informacije, čime se sprječava kolaps modova.







4. Rezultati

Training images (Highest loss)		Accompanied text description	Training images (Lowest loss)		Accompanied text description
Real	Generated		Real	Generated	
		the bird has an orange crown and a small black bill. a medium sized bird with a red crescent on...			the bird is gray and the head of the bird is white with the beak of the bird being yellow and short. the bird has a...
Loss: 313.62			Loss: 9.67		
		this bird has a white belly with a long and pointy orange beak. this is a white bird with black wings...			this smaller bird has a curved beak and green wings. the bird has a green...
Loss: 295.99			Loss: 9.63		
		this is an all black bird with a large black beak and black feet. this bird is completely black, including...			a grey bird that has a lighter belly and orange beady eyes with a slight mohawk. this is a bird...
Loss: 301.31			Loss: 10.73		
		yellow underbelly with grey wings and a long pointy beak. this is a bird with a yellow breast...			a large bird has a head with a black crown, a small and sharp bill, and a back with tufts of brown feathers..
Loss: 339.16			Loss: 10.75		

Slika 4.1. Podskup slika korištenih za treniranje

Prilikom treniranja DCGAN modela na skupu slika ptica, uočena je manja greška na slikama gdje se ptice stapaju s okolinom. Ova pojava može biti povezana s načinom na koji BCE loss (binarna unakrsna entropija) funkcionira u kontekstu GANova. DCGAN-ov generator teži minimiziranju greške tako da "prevrati" diskriminator, no izražene konture i jasno odvojene ptice od pozadine zahtijevaju preciznije generiranje detalja. Kada se ptice stapaju s okolinom, modelu je lakše generirati manje kontrastne teksture, što rezultira nižom greškom. S druge strane, istaknute ptice zahtijevaju točniju reprodukciju

geometrijskih oblika i kontura, što može dovesti do nestabilnosti u treniranju i nasumičnih sivih područja.

Testing images		Accompanied text description	Generated by an overfitted model (740 epochs)
Real	Generated		
		this bird has long black legs, brown feathers, and a black beak. this bird is mostly brown with a short pointy bill...	
		small to medium sized grey bird with black wings long beak and long tarsus this bird is white and black in color with a long skinny black beak and black eye ring...	
		this is a dark grey bird with a white wing and a black beak. this bird is covered in all black feathers all over its body except for its coverts which are white. this black bird has white..	
		the bird has a small bill that is black and black feet. as the bird stands on a branch its orange body and navy blue feathers are distinct bird has gray body feathers, bronze breast feather, ...	

Slika 4.2. Podskup slika korištenih za testiranje

U kontekstu generiranja tekstualnih opisa, poteškoće u postizanju zadovoljavajućih rezultata mogu biti posljedica stilske ovisnosti BERT enkodera. BERT enkoderi, iako dobri u hvatanju semantičkih veza, često usvajaju stilsku strukturu iz trening podataka. Ako korisnički upit (prompt) ne odgovara stilu i sintaksi korištenoj u originalnom datasetu, enkoder može generirati loše prilagođene reprezentacije. Na primjer, ako je dataset tekstova koristio formalne opise, neformalni upit može dovesti do neskladnih vektorskih reprezentacija koje otežavaju generatoru pravilnu interpretaciju. Ova pojačana osjetljivost na stil zahtijeva fino podešavanje enkodera ili korištenje metoda poput "prompt engineeringa" za bolju usklađenost.

5. Zaključak

U ovom radu uspješno je implementiran i evaluiran DCGAN model za generiranje realističnih slika ptica na temelju tekstualnih opisa, koristeći napredne tehnike dubokog učenja i alate poput PyTorch-a. Korištenje BERT enkodera omogućilo je učinkovitu pretvorbu složenih jezičnih opisa u vektorske reprezentacije, čime je poboljšana semantička povezanost između teksta i generirane slike.

Rezultati eksperimenta pokazuju da pažljivo dizajnirana arhitektura generatora i diskriminatora, uz primjenu višestrukih funkcija gubitka (BCE, L1, L2), omogućuje modelu da postigne ravnotežu između realističnosti generiranih slika i njihove vjernosti ulaznim opisima. Uvođenje pogrešnih stvarnih slika u treniranje dodatno je povećalo robusnost diskriminatora, čime je spriječen kolaps modova i povećana raznolikost generiranih uzoraka.

Analiza utjecaja upravljanja stopom učenja pokazala je da konvencionalni rasporedi stope učenja nisu optimalni za GAN arhitekture, jer mogu dovesti do preuranjene konvergencije i suboptimalnih rezultata. Najbolje performanse postignute su korištenjem fiksne stope učenja, što je omogućilo stabilnu dinamiku između generatora i diskriminatora tijekom treniranja. Vizualni i kvantitativni rezultati potvrđuju sposobnost modela da generira slike koje su semantički usklađene s tekstualnim opisima.

Zaključno, implementirani sustav demonstrira potencijal dubokih generativnih modela u sintezi slika iz teksta, otvarajući mogućnosti za daljnja istraživanja i primjene u računalnom vidu, automatiziranoj ilustraciji te proširenju na druge domene i kompleksnije opise. Osim DCGAN pristupa korištenog u implementaciji, rad opisuje i druge metode generiranja slika iz teksta, pružajući širu perspektivu i kontekstualizaciju postignutih rezultata unutar postojećih istraživačkih okvira.

Literatura

- [1] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, i H. Lee, “Generative adversarial text to image synthesis”, u *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016., str. 1060–1069. [Mrežno]. Adresa: <https://arxiv.org/abs/1605.05396>
- [2] M. A. Team, “An introduction to diffusion models and stable diffusion”, Marvik AI Blog, November 2023., 28. studenoga 2023. [Mrežno]. Adresa: <https://blog.marvik.ai/2023/11/28/an-introduction-to-diffusion-models-and-stable-diffusion/>
- [3] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, i B. Ommer, “High-resolution image synthesis with latent diffusion models”, u *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022., str. 10 684–10 695. [Mrežno]. Adresa: <https://arxiv.org/abs/2112.10752>
- [4] D. Surve, “Step-by-step guide to implement latent diffusion”, Kaggle Notebook, July 2023., 29 Srpnja 2023. [Mrežno]. Adresa: <https://www.kaggle.com/code/deveshsurve/step-by-step-guide-to-implement-latent-diffusion>
- [5] Baeldung, “From rnns to transformers”, Baeldung on Computer Science, March 2023., 18. ožujka 2023. [Mrežno]. Adresa: <https://www.baeldung.com/cs/rnns-transformers-nlp>
- [6] J. Ferrer, “How transformers work: A detailed exploration of transformer architecture”, DataCamp Tutorial, January 2024., 9. siječnja 2024. [Mrežno]. Adresa: <https://www.datacamp.com/tutorial/how-transformers-work>
- [7] K. P. Lata, “Understanding the encoder-decoder transformer: A deep dive”, LinkedIn Pulse, September 2024., 25. rujna 2024. [Mrežno]. Adresa:

<https://www.linkedin.com/pulse/understanding-encoder-decoder-transformer-deep-dive-kumar-preeti-lata-zc2te>

- [8] IBM, “What is a transformer model?” IBM Think Blog, October 2023., 10. listopada 2023. [Mrežno]. Adresa: <https://www.ibm.com/think/topics/transformer-model>
- [9] S. Kumar, “How transformer models generate images from text”, LinkedIn Pulse, April 2025., 7. travnja 2025. [Mrežno]. Adresa: <https://www.linkedin.com/pulse/how-transformer-models-generate-images-from-text-santosh-kumar-az2oc>
- [10] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, i I. Sutskever, “Zero-shot text-to-image generation”, *arXiv preprint*, February 2021., 24. veljače 2021. [Mrežno]. Adresa: <https://arxiv.org/abs/2102.12092>
- [11] R. Hada, “Text to photo llm: Revolutionizing visual generation with ai”, Future AGI Blog, April 2025., 16. travnja 2025. [Mrežno]. Adresa: <https://futureagi.com/blogs/text-to-photo-llm-revolutionizing-visual-generation-with-ai>
- [12] M. Magat, “Dcgan-text-to-bird-image”, <https://github.com/matejmagat/DCGAN-text-to-bird-image>, 2024., gitHub repository.
- [13] <https://pytorch.org/>.
- [14] P. Team, “Dcgan tutorial - generative adversarial networks”, https://docs.pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html, 2024., accessed: 2025-05-22.
- [15] A. Radford, L. Metz, i S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks”, *arXiv preprint arXiv:1511.06434*, 2015. [Mrežno]. Adresa: <https://arxiv.org/abs/1511.06434>
- [16] H. Hazimeh i N. Ponomareva, “Mind the (optimality) gap: A gap-aware learning rate scheduler for adversarial nets”, u *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2023., str. 11 044–11 062. [Mrežno]. Adresa: <https://proceedings.mlr.press/v206/hazimeh23a/hazimeh23a.pdf>