SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Duboko učenje 2: Izvještaj projekta

GAN sustav za generiranje slika automobila

Igor Aradski Robert Travančić Vilim Lendvaj
Ivan Lokas Luko Lulić Josip Vucić

Zagreb, siječanj 2023.

SADRŽAJ

1. Uvod u temu

U ovom projektu istražit ćemo korištenje Generative Adversarial Network (GAN) algoritma za generiranje slika automobila. U ovom poglavlju slijedi kratak uvod u algoritam.

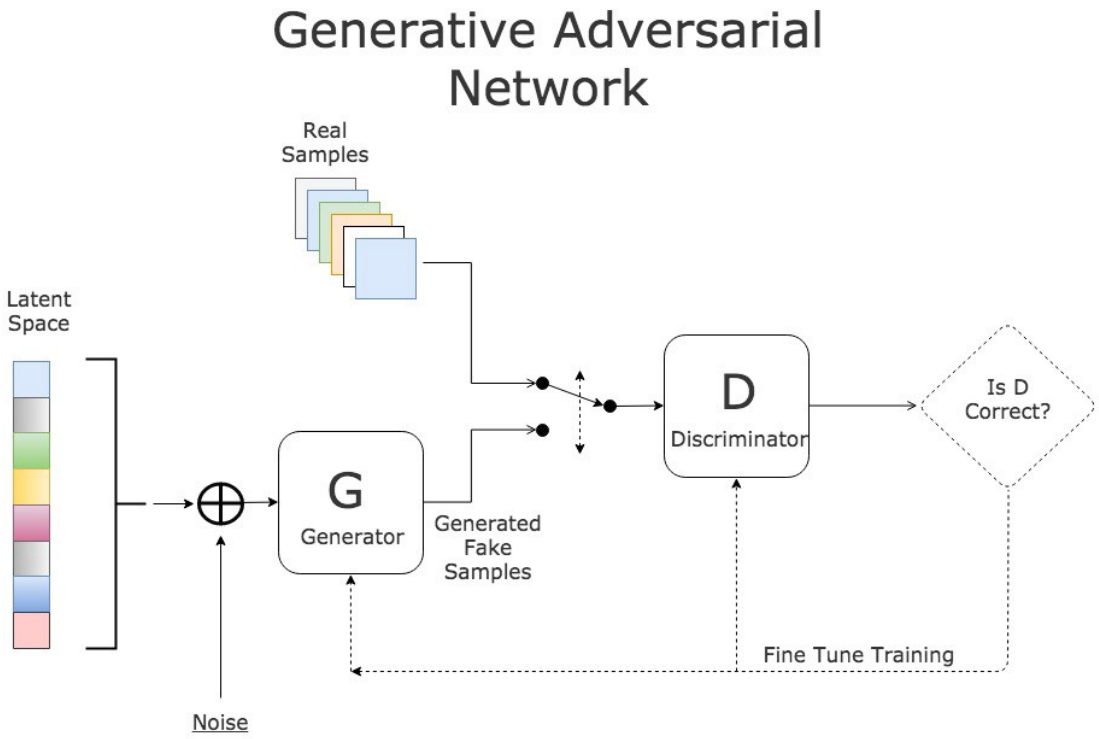
1.1. Generative adversarial network(GAN)

Generative adversarial network (GAN) su vrsta algoritama dubokog učenja koji su posljednjih godina privukli značajnu pozornost zbog svoje sposobnosti generiranja sintetičkih podataka koji su vrlo realistični. Algoritam je predstavljen 2014. godine te mu je autor Ian Goodfellow [1]. Korišteni su za različite zadatke, uključujući generiranje slika, generiranje teksta i generiranje glazbe.

Slika 1.1: *Ian Goodfellow* [1]

1.2. Struktura modela

Struktura GAN-a sastoji se od dvije neuronske mreže: generatora i diskriminatora. Generator je odgovoran za generiranje sintetičkih podataka, dok je diskriminator odgovoran za razlikovanje stvarnih od sintetičkih podataka.Generator je tipično dekonvolucijska neuronska mreža, što znači da povećava uzorkovanje ulaznih podataka niže razlučivosti da proizvede izlazne podatke više razlučivosti. Na primjer, u slučaju generiranja slike, generator može uzeti nasumični vektor šuma i generirati sintetičku sliku koja se ne može razlikovati od stvarne slike. Diskriminator je obično konvolucijska neuronska mreža, što znači da obrađuje ulazne podatke primjenom skupa filtara i smanjivanjem uzorkovanja podataka kako bi proizveo izlaz niže razlučivosti. U slučaju generiranja slike, diskriminator može uzeti sliku i ispisati vjerojatnost koja pokazuje koliko je vjerojatno da je slika stvarna.



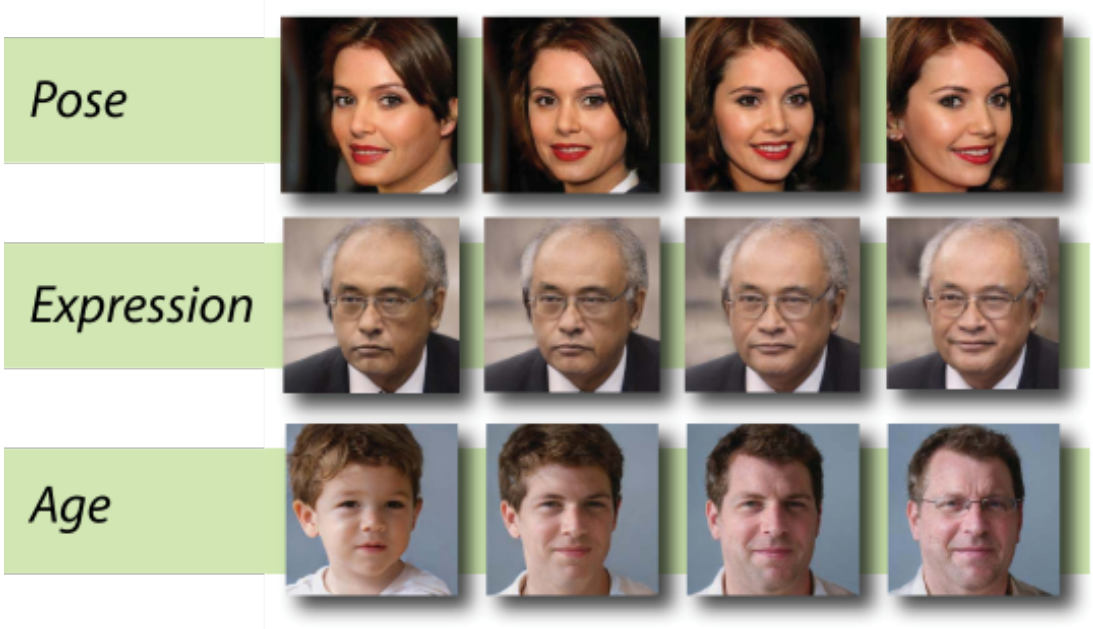
Slika 1.2: *Struktura* [2]

Proces obuke GAN-ova uključuje generator i diskriminator koji se zajedno treniraju na suparnički način. Generator proizvodi seriju sintetičkih podataka, koja se zatim ubacuju u diskriminator zajedno sa serijom stvarnih podataka. Diskriminator obrađuje obje serije i ispisuje vjerojatnost koja pokazuje koliko je vjerojatno da je svaki podatak stvaran. Generator i diskriminator ažuriraju se na temelju performansi diskriminatora. Ako diskriminator može ispravno identificirati većinu sintetičkih podataka kao lažne,

tada se generator ažurira kako bi proizveo bolje sintetičke podatke. S druge strane, ako diskriminator ima problema s razlikovanjem pravih i sintetičkih podataka, ažurira se kako bi bolje identificirao lažne podatke. Jedan od ključnih izazova u obuci GAN-ova je osigurati da i generator i diskriminator mogu napredovati. Ako generator postane previše dobar u stvaranju sintetičkih podataka, diskriminator bi mogao imati problema s identificiranjem lažnih podataka, što dovodi do zastoja u učenju. Slično, ako diskriminator postane predobar u identificiranju lažnih podataka, tada se generator može mučiti s proizvodnjom sintetičkih podataka koje nije lako identificirati kao lažne.

Također je poznato da GAN-ovi pate od kolapsa načina rada, gdje generator proizvodi samo ograničeni raspon izlaza, a ne raznolik raspon. To može biti uzrokovano zaglavljivanjem generatora i diskriminatora u lokalnom minimumu, gdje generator proizvodi sintetičke podatke koji su dovoljno dobri da zavaraju diskriminator, ali nisu dovoljno raznoliki da zahvate cijeli niz stvarnih podataka.

Općenito, GAN-ovi su se pokazali solidnim izborom na raznim zadacima, a mnogi su postigli i impresivne rezultate koristeći ih. Međutim, još uvijek ima prostora za poboljšanje i potrebna su daljnja istraživanja kako bi se bolje razumjelo kako učinkovito procijeniti i optimizirati ovaj tip algoritma.



Slika 1.3: *Generirane slike* [3]

2. Model

2.1. Model za generiranje slika automobila

Cilj ovog modela je uspješno generirati slike realističnih automobila. Za treniranje modela koristili smo Stanford cars dataset. Dataset sadrži više od 16000 slika automobila koji su podijeljeni u 196 klasa te uključuje podatke o boji, marki, modelu i godini za svaku klasu.

2.2. Arhitektura rješenja

Arhitektura za koju smo dobili dobre rezultate nalazi se u nastavku.

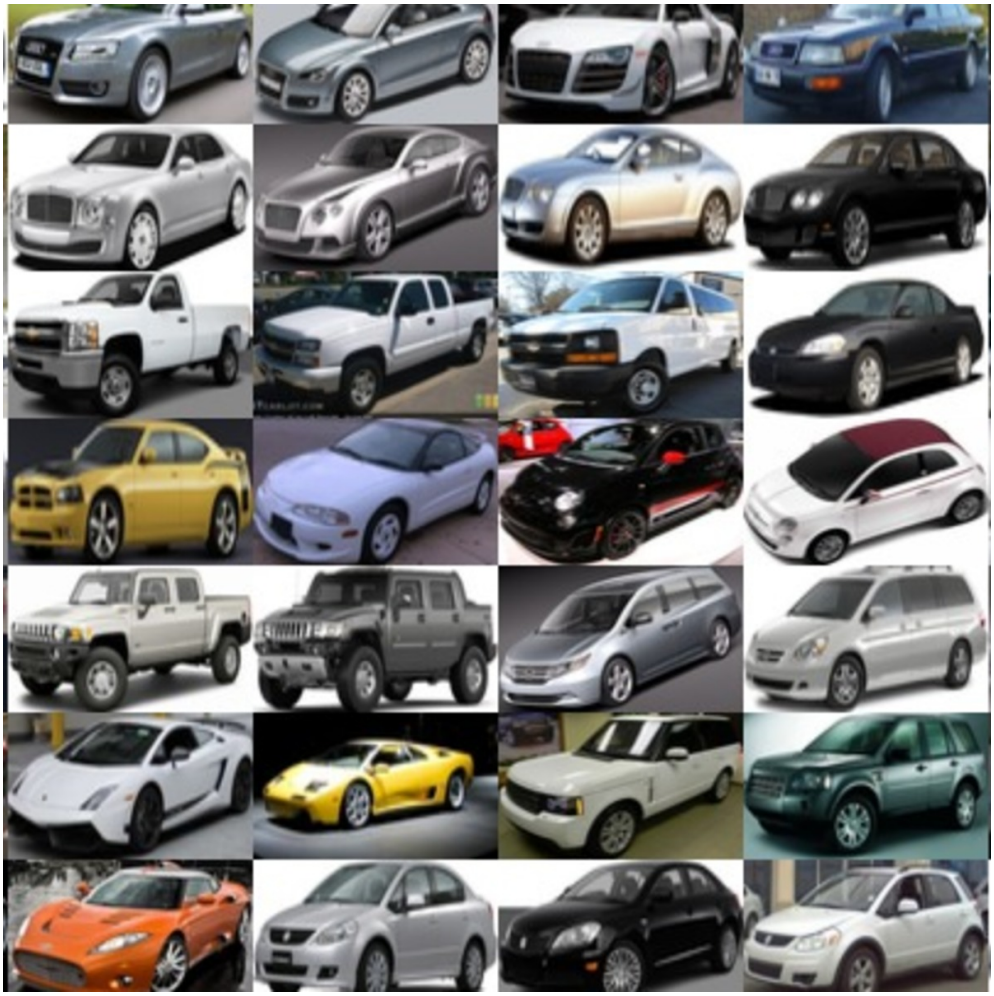
Generator:

- 7 transponiranih konvolucijskih slojeva (1024 x ... X 32)
- Jezgra veličine 4
- Batch normalizacija
- Leaky ReLU prijenosna funkcija

Diskriminator:

- 7 konvolucijskih slojeva (32 x ... X 1024)
- Jezgra veličine 4
- Batch normalizacija
- Leaky ReLU prijenosna funkcija

U nastavku se nalazi kod korišten u implementaciji modela.



Slika 2.1: *S*tanford cars dataset

Listing 2.1: Python implementacija modela

```
import torch.nn as nn


class ConvolutionalDiscriminator(nn.Module):
    def __init__(self, image_size, n_channels):
        super().__init__()

        self.image_size = image_size
        self.n_channels = n_channels

        self.layers = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(32),
            nn.LeakyReLU(0.02, inplace=True),

            nn.Conv2d(32, 64, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(64),
            nn.LeakyReLU(0.02, inplace=True),
```

```
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.02, inplace=True),

            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.02, inplace=True),

            nn.Conv2d(256, 512, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.02, inplace=True),

            nn.Conv2d(512, 1024, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(1024),
            nn.LeakyReLU(0.02, inplace=True),

            nn.Conv2d(1024, 1, kernel_size=4, stride=1, padding=0, bias=False),

            nn.Flatten(),
            nn.Sigmoid(),
        )

    def forward(self, x):
        return self.layers(x)


class ConvolutionalGenerator(nn.Module):
    def __init__(self, image_size, noise_dimension):
        super().__init__()

        self.image_size = image_size
        self.noise_dimension = noise_dimension

        self.layers = nn.Sequential(
            nn.ConvTranspose2d(noise_dimension, 1024, kernel_size=4, stride=1, padding=0, bias=False),
            nn.BatchNorm2d(1024),
            nn.LeakyReLU(0.02, inplace=True),

            nn.ConvTranspose2d(1024, 512, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.02, inplace=True),

            nn.ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.02, inplace=True),

            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.02, inplace=True),

            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(64),
            nn.LeakyReLU(0.02, inplace=True),
```

```
            nn.ConvTranspose2d(64, 32, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(32),
            nn.LeakyReLU(0.02, inplace=True),

            nn.ConvTranspose2d(32, 3, kernel_size=4, stride=2, padding=1, bias=False),
            nn.Tanh()
        )

    def forward(self, x):
        return self.layers(x)
```

Klasa ConvolutionalDiscriminator definira konvolucijsku neuronsku mrežu (CNN) koja prima sliku i ispisuje vrijednost između 0 i 1 koja predstavlja vjerojatnost da je slika stvarna. CNN se sastoji od nekoliko konvolucijskih slojeva, nakon kojih slijedi sloj batch normalizacije i funkcija aktivacije Leaky ReLU. Posljednji sloj je sigmoidna aktivacijska funkcija koja proizvodi izlaz koji predstavlja vjerojatnost.

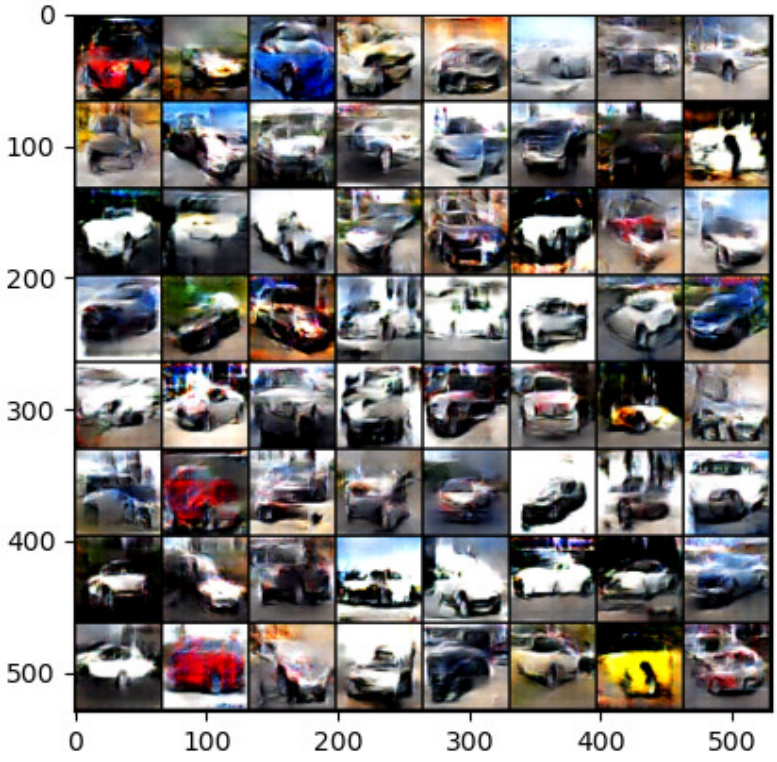
Klasa ConvolutionalGenerator definira CNN koji uzima vektor šuma i generira sintetičku sliku. CNN se sastoji od nekoliko transponiranih konvolucijskih slojeva, nakon kojih svaki slijedi batch normalizacijski sloj i funkcija aktivacije Leaky ReLU. Završni sloj je Tanh aktivacijska funkcija koja proizvodi generiranu sliku.

Treniranje se izvodi u 100 epoha, latentna dimenzija je 128 te dimenzije slika su 256x256. Trajanje treniranja je bilo oko 4 sata.

2.3. Rezultati

Uz model napravljen je i interaktivni GUI koji poboljšava iskustvo korištenja modela. Slike koje je generator naučio generirati nalaze se u nastavku. Vidimo da je model uspio naučiti neke karakteristike koje posjeduju slike iz dataseta na kojem je treniran kao što su oblik, postojanje kotača, guma, prozora i slično.

Postoji prostor za daljnje unapređenje modela kao što su ispitivanje većeg skupa hiperparametara te drugih konfiguracija arhitekture.



Slika 2.2: *G*enerirana slika

Slika 2.3: *Generirana slika*

LITERATURA

[1] https://www.techspot.com/news/94512-apple-loses-director-machine-learning-over-return-person.html

[2] https://paperswithcode.com/method/gan

[3] https://www.arxiv-vanity.com/papers/1909.06122/

[4] https://en.wikipedia.org/wiki/Generative_adversarial_network

[5] https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

[6] https://snap.stanford.edu/data/