Generisanje slike lica
GAN

Generative Adversarial Networks

Autori

Natalija Jovanović 226/2015
Marija Milićević 105/2015

Matematički fakultet
Februar 2019.

Uvod

Veštačke neuronske mreže su familija statističkih modela učenja inspirisana biološkim neuronskim mrežama. Koriste se u svrhu aproksimacije funkcija koje mogu zavisiti od velike količine ulaznih podataka, a koje su u principu nepoznate. Veštačke neuronske mreže su sistemi medjusobno povezanih neurona koji šalju poruke jedni drugima. Veze izmedju ovih neurona imaju numeričke težine koje mogu biti podložne promenama u zavisnosti od iskustva, što neuronske mreže čini adaptivnim i sposobnim za učenje.

GAN su vrsta neuronskih mreža koje se sastoje od dve neuronske mreže koje se medjusobno takmiče. Ova tehnika može da generiše slike tako da posmatraču izgledaju barem površno autentično time što imaju dosta realnih karakteristika.

Opis problema

Glavni cilj GAN-a je da izgeneriše podatke od nule (uglavnom slike, ali moze i muziku). GAN pravi dve mreže - generator, koji generiše podatke, i diskriminator, koji procenjuje da li je podatak dosao iz pravih podataka ili ga je generator generisao (binarni klasifikator). GAN trenira te mreže koje se takmiče sve dok se ne uspostavi neka ravnoteža. Cilj je da se obe mreže vremenom poboljšavaju, što prouzrokuje da generator da što realniji izlaz. Ova situacija može da se modeluje minimax igrom. Dakle, postoje dva agenta, kriminalac i policajac. Ciljevi su im suprotni, kriminalac hoće da na kompleksne načine falsifikuje novac, tako da policajac ne može da prepozna da li je novac falsifikovan ili ne, dok policajac hoće da što bolje prepozna da li je novac falsifikovan ili ne. Tokom procesa i policajac i kriminalac razvijaju sve sofisticiranije tehnologije da ispune svoje ciljeve.

Prvo uzmemo neki šum z iz normalne ili uniformne raspodele. Sa z kao ulaz, koristimo generator G da napravimo sliku x, koja je slična skupu podataka koji imamo ($x = G(z)$). Konceptualno, z predstavlja skrivene karakteristike generisane slike (npr. boju, oblik). Dakle, generator je funkcija koja transformiše slučajan ulaz u sintetički izlaz, gde je slučajan ulaz neki 2D uzorak sa vrednošću (x, y), a izlaz je 2D uzorak koji je mapiran na drugu poziciju, tj. lažni uzorak.

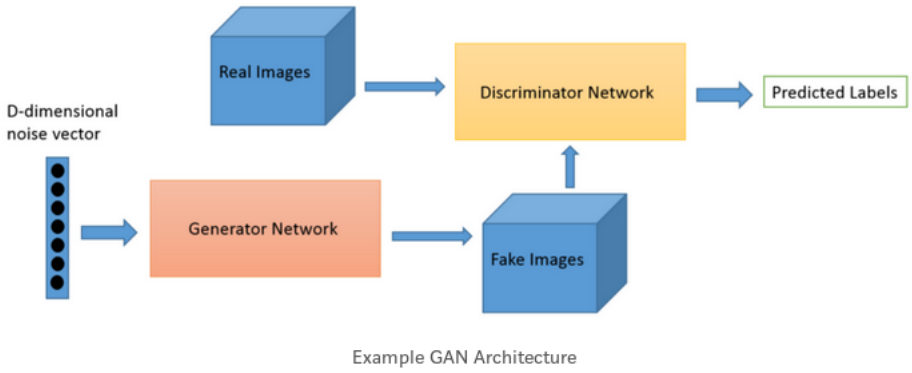

Figure 1: GAN

Zašto uopšte koristiti šumove? Ne bi bilo zanimljivo da se napravi sistem koji daje isti rezultat svaki put kad se pokrene. Takodje je bitno razmišljati o verovatnoćama, jer nam pomažu da prebacimo problem generisanja slike u "prirodnu matematiku".
Zbog modelovanja funkcije u višedimenzionom prostoru su neuronske mreže i napravljene.

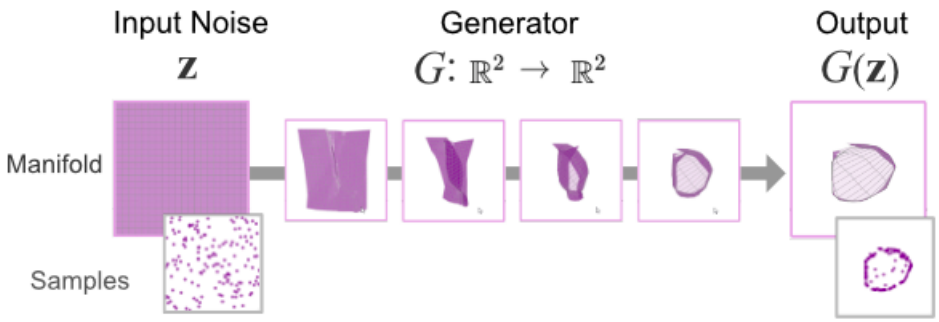

Figure 2: Generisanje slike uz pomoć šumova

Funkcija gubitka

Diskriminator za izlaz daje vrednost $D(x)$, tj. verovatnoću da je x iz realnih slika. Mi želimo da maksimizujemo verovatnoću da tačno prepozna realnu sliku kao realnu, generisanu kao lažnu. Kao merilo gubitka, koristišćena je mera cross-entropy ($plog(p)$).

$$\max_D V(D) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

Takodje, želimo da generator što bolje prevari diskriminatora (minimizuje se verovatnoća da prepozna lažnu sliku).

$$\min_G V(G) = E_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

Kada su obe funkcije definisane, zajedno uče pomoću gradijentnog spusta. Popravljamo parametre generatornog modela i izvodimo jednu iteraciju gradijentnog spusta nad diskriminatorom koristeći prave i generisane slike. Zatim menjamo stranu. Popravljamo diskriminator i treniramo generator za jednu iteraciju. Treniramo obe mreže naizmeničnim koracima sve dok generator ne proizvede kvalitetnu sliku. Gradijenti se koriste za propagaciju unazad.

$$\min_G \max_D V(D, G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

Konvolutivne neuronske mreže

Konvolutivne neuronske mreže primenjuju serije filtera na sirove podatke slika kako bi izdvojile i naučile karakteristike višeg nivoa, koje model kasnije koristi za klasifikaciju.
Regularne neuronske mreže za ulaz dobijaju jedan vektor i transformišu ga kroz niz skrivenih slojeva. Kada su slike dimenzija 64x64x3 (64 dužina, 64 visina, 3 kanala za boju), prvi skriveni sloj bi imao $64 * 64 * 3 = 12288$ težina. Ovaj broj deluje izvodljivo, ali očigledno ova potpuno povezana struktura ne može da skalira do većih slika. Kod slika realnijih veličina, npr. 200x200x3, neuroni će imati $200 * 200 * 3 = 120000$ težina. Štaviše, skoro sigurno bismo želeli da imamo nekoliko takvih neurona tako da bi se parametri brzo uklopili. Očigledno, ova potpuna povezanost je rasipna i veliki broj parametara bi brzo doveo do preprilagodjenosti. Za razliku od običnih neuronskih mreža, sloj konvolutivne neuronske mreže uredjuje neurone u tri dimenzije: širina, visina, dubina. Neuroni će biti povezani samo sa malim regijama iz prethodnog sloja umesto sa svim neuronima.

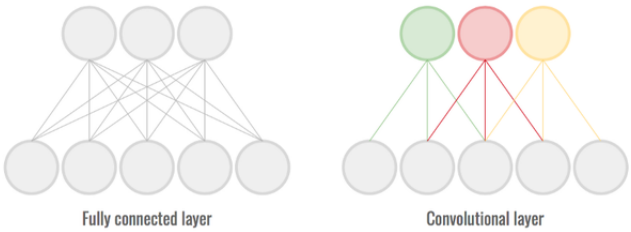

Figure 3: Razlika izmedju regularne neuronske mreže i konvolutivne neuronske mreže

Konvolutivne neuronske mreže se sastoje iz pet delova:

1. Ulaz - sadrži sirove podatke slike

2. Konvolutivni sloj - primenjuje odredjeni broj konvolutivnih filtera na sliku; za svaku podoblast, sloj izvršava skup matematičkih operacija kako bi dobio jednu vrednost na izlazu.

3. *ReLU* sloj - primenjuje *ReLU* aktivacionu funkciju (*ReLU* - pozitivni deo argumenta, $f(x) = max(0, x), x$ je ulaz).

4. Slojevi za objedninjavanje - upsample-uje podatke slike, izdvajajući ih po konvolutivnim slojevima da bi smanjio dimenzionost mape karakteristika kako bi se smanjilo vreme procesiranja. Algoritam koji se koristi je *maxpooling* koji izdvaja regione mape karakteristika (npr. 2x2 piksel ploča), čuva njihovu maksimalnu vrednost i odbacuje sve ostale vrednosti.

5. Slojevi gustine (potpuno povezani) - izvršavaju klasifikaciju nad karakterisikama koje su izdvojene konvolutivnim slojem i upsample-ovane u sloju objedinjavanja. Svaki čvor iz sloja je povezan sa svakim čvorom iz sloja predikcije.

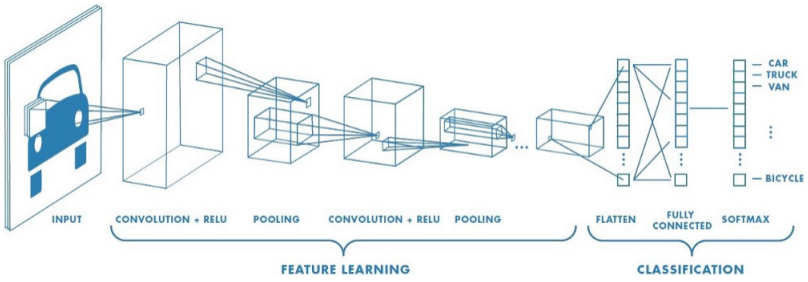

Figure 4: Konvolutivna neuronska mreza

Dobijeni rezultati

GAN smo implementirale u python3.7 koristeći keras biblioteku. I generator i diskriminator su konvolutivne neuronske mreže. Mreže su trenirane na CPU-u (vreme treniranja oko 36h, bilo bi manje da je trenirano na GPU-u). Specifikacije sistema na kome je vršeno treniranje:
CPU: Intel Core i5-5200U CPU @ 2.20GHz 4
GPU: Intel HD Graphics 5500 (Broadwell GT2)
RAM: 7.7GiB
OS: Ubuntu 18.04.1 LTS
OS tip: 64-bit

Implementacija

Napravile smo klasu GAN koja kao podatke ima oblik slika (vrste, kolone, kanali), diskriminator i generator. Instancu te klase smo trenirale tako što smo za svaku epohu radile sledeće:

- Pošto slika ima dosta (preko 200000), njihov skup smo podelile na manje celine nad kojima je kasnije vršeno treniranje
- Diskriminator - za svoj skup, koji je definisan veličinom *batch_size* (u našem slučaju je 256), dobija pola realnih i pola generisanih slika i pokusava da utvrdi koja slika dolazi iz kog skupa
- Generator - želi da diskriminator kategoriše generisane slike kao realne i trenira

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 4096)              16781312
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 4096)              0
_________________________________________________________________
batch_normalization_1 (Batch (None, 4096)              16384
_________________________________________________________________
reshape_1 (Reshape)          (None, 16, 16, 16)        0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 32, 32, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 128)       51328
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 128)       512
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 3)         9603
=================================================================
Total params: 16,859,139
Trainable params: 16,850,691
Non-trainable params: 8,448
```

Figure 5: Generator

```
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 12288)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               6291968
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 6,423,553
Trainable params: 6,423,553
Non-trainable params: 0
```

Figure 6: Diskriminator

Rezultati

```
0 1 [D loss: 0.581309, acc.: 63.28%] [G loss: 0.870803]
256
0 2 [D loss: 0.365933, acc.: 81.25%] [G loss: 1.267071]
256
0 3 [D loss: 0.236191, acc.: 89.84%] [G loss: 1.624951]
256
0 4 [D loss: 0.173498, acc.: 93.75%] [G loss: 1.770371]
256
0 5 [D loss: 0.289378, acc.: 88.67%] [G loss: 1.643906]
256
0 6 [D loss: 0.171729, acc.: 95.70%] [G loss: 1.932828]
256
0 7 [D loss: 0.142237, acc.: 96.09%] [G loss: 2.126240]
256
0 8 [D loss: 0.120291, acc.: 98.05%] [G loss: 2.547749]
256
```

Figure 7: Rezultati na početku treniranja

```
3 465 [D loss: 0.494450, acc.: 74.61%] [G loss: 1.747836]
256
3 466 [D loss: 0.488027, acc.: 76.56%] [G loss: 1.819122]
256
3 467 [D loss: 0.473183, acc.: 75.00%] [G loss: 1.743265]
256
3 468 [D loss: 0.485808, acc.: 75.39%] [G loss: 1.731369]
256
3 469 [D loss: 0.502168, acc.: 75.78%] [G loss: 1.783053]
256
3 470 [D loss: 0.483663, acc.: 76.56%] [G loss: 1.847780]
256
3 471 [D loss: 0.504533, acc.: 77.73%] [G loss: 1.997403]
256
3 472 [D loss: 0.563566, acc.: 73.44%] [G loss: 1.804208]
256
```

Figure 8: Rezultati na sredini treniranja

Može se primetiti da tokom treniranja preciznost diskriminatora opada, a generator se trudi da mu gubitak bude što manji. To je i očekivano, jer generator treba što bolje da izgeneriše sliku da diskirminator sa skoro jednakom verovatnoćom odredi da li je slika realna ili lažna.

```
5 782 [D loss: 0.544736, acc.: 71.48%] [G loss: 1.947613]
256
5 783 [D loss: 0.509805, acc.: 76.17%] [G loss: 1.688098]
256
5 784 [D loss: 0.508034, acc.: 75.39%] [G loss: 1.573433]
256
5 785 [D loss: 0.506062, acc.: 74.61%] [G loss: 1.515670]
256
5 786 [D loss: 0.554161, acc.: 71.09%] [G loss: 1.422990]
256
5 787 [D loss: 0.486395, acc.: 76.95%] [G loss: 1.595928]
256
5 788 [D loss: 0.552882, acc.: 71.88%] [G loss: 1.581754]
256
5 789 [D loss: 0.508250, acc.: 73.83%] [G loss: 1.592129]
256
5 790 [D loss: 0.461168, acc.: 75.78%] [G loss: 1.594994]
Saved model to disk
```

Figure 9: Rezultati na kraju treniranja

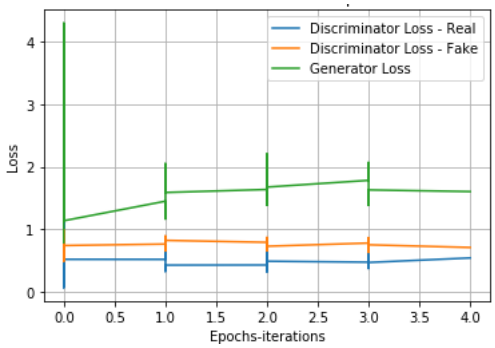

Figure 10: Funkcije gubitka

Tokom procesa treniranja izdvajaju se generisane slike i može se primetiti da se vremenom tokom epoha izdvajaju siluete lica. Slike nisu kristalne i ima dosta šumova, ali to je do broja epoha.

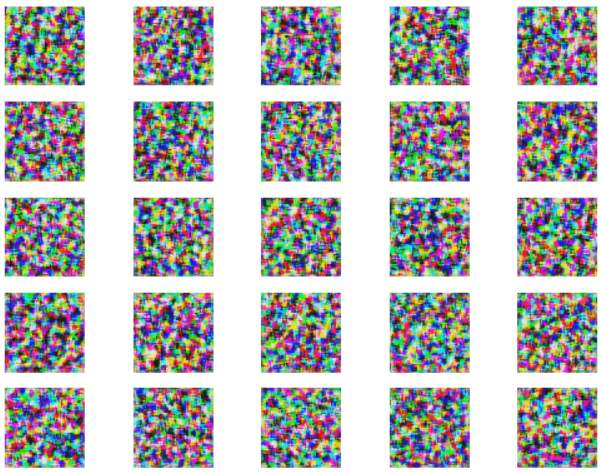

Figure 11: Slike na početku treniranja

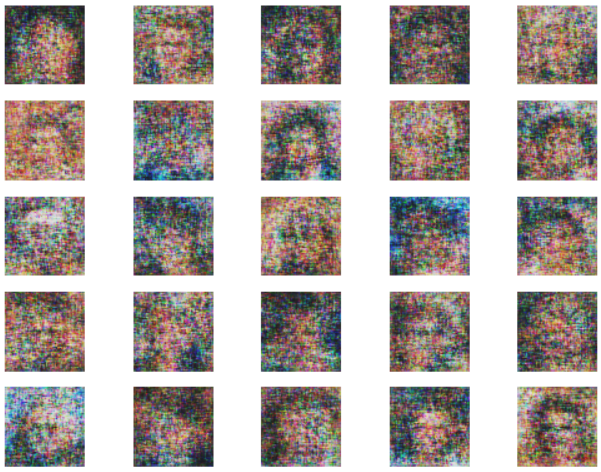

Figure 12: Slike na početku druge epohe

Figure 13: Slike na početku treće epohe

Figure 14: Slike na početku četvrte epohe

Figure 15: Slike na početku pete epohe

Figure 16: Slike na početku šeste epohe

Figure 17: Slike na kraju šeste epohe

Zaključak

Rezultati koje smo dobile nisu bas sjajni, odnosno ima dosta šumova. Medjutim, mozemo primetiti kako se broj epoha povećava da se tako povećava i kvalitet slika. Pošto se mreža trenira dosta dugo, nismo probale sa više od 6 epoha, ali da jesmo sigurno bismo dobile bolje i čistije slike.

Literatura

[1] Ian J. Goodfellow, Jean Pouget-Abadie , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio *Generative Adversarial Nets*

[2] Peter J.B. Hancock and Charlie D. Frowd *Evolutionary generation of faces*

[3] Jon Gauthier *Conditional generative adversarial nets for convolutional face generation*

[4] Yaohui Wang, Antitza Dantcheva, Francois Bremond *From attribute-labels to faces: face generation using a conditional generative adversarial network*

[5] *https://fairyonice.github.io/My-first-GAN-using-CelebA-data.html*

[6] *https://towardsdatascience.com/having-fun-with-deep-convolutional-gans-f4f8393686ed*

[7] *https://poloclub.github.io/ganlab/*

[8] *https://github.com/eriklindernoren/Keras-GAN/tree/master/gan*