13) Intro to GANs

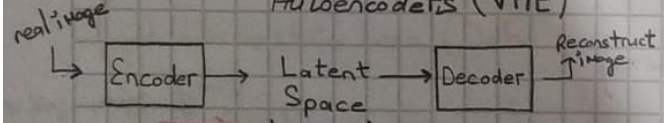
→ Unsupervised Technique.

Discriminative Models	Generative Models
✗ Used for Classification in ML	✗ Generate new data instances
P(Y|X) bulmaya çalışır	Capture P(X|Y), or just P(X) if there are no labels.

There are many types of **Generative Models**.

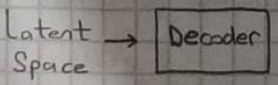
Aşağıda en çok kullanılan ikisini görüyoruz.

1) Variational Autoencoders (VAE)

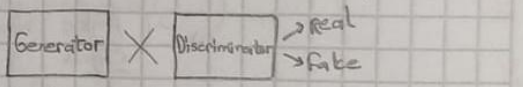


Representation of compressed data.
(Linkini PC'ye kaydettik "Gans" folder altında)

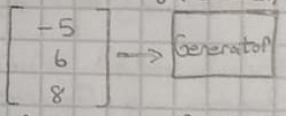
→ After training, encoder part is removed and Latent vector is used for generating new images.



2) Generative Adversarial Networks (GAN)



→ Next step: (Eğitim bitince Discriminatore ihtiyaç kalmaz.)



Özetle:

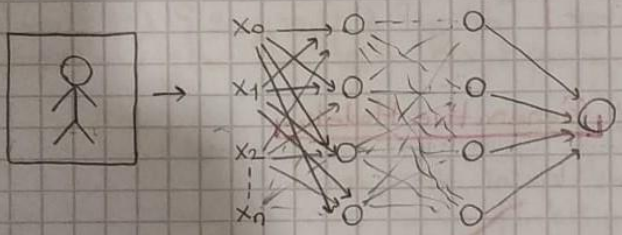
- Discriminator decides whether an input image is real or fake.
- Generator takes random noise and try to generate realistic image from the Learnt distribution of the training data.

Discriminator :

Goal : Distinguish between different classes.

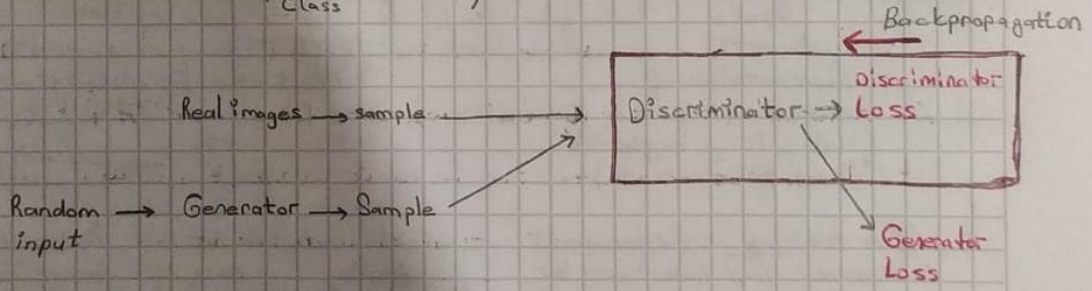
Modeling the probability of class Y given input features X.

↳ $P(Y|X)$
Class Features



⇒ Models the probability of an example being fake given a set of inputs X.

↳ $P(\text{Fake Class} \mid \text{features})$



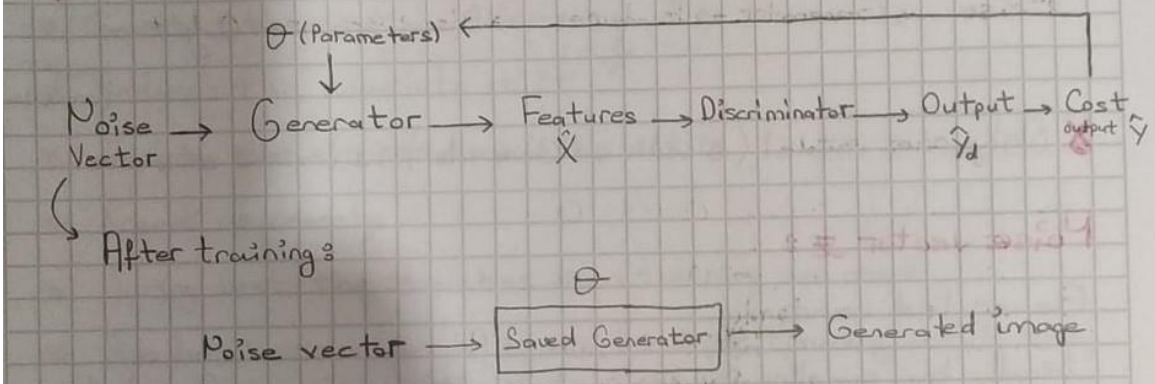
!!! During Discriminator Training the Generator does not Train.

!!! Discriminator Loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.

!!! Discriminator updates its weights through "Backpropagation" from the discriminator Loss through the discriminator network.

!!! Similarly, we keep the discriminator constant during the Generator Training phase.

Generator :



After training :

Loss Functions :

1) Minimax loss (Derived from Cross-Entropy)
2) Wasserstein Loss (Daha sonra incelenecek)

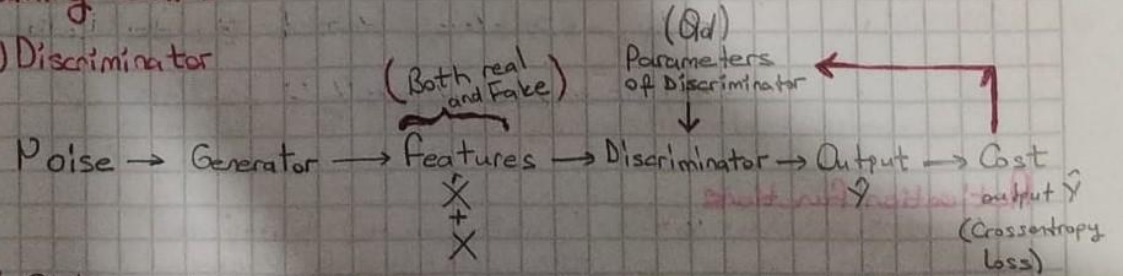
A GAN has 2 Loss functions: One for generator and one for discriminator training. But they come from a single formula.

Minimax Loss ⇒ Generator tries to minimize below function;
Discriminator tries to maximize it.

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

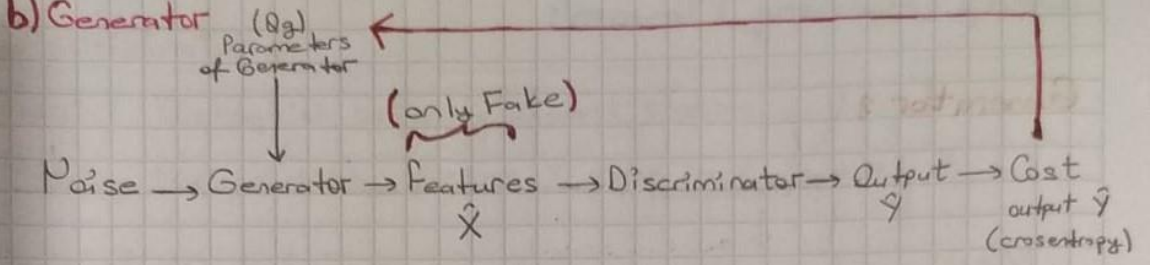
Training GANs :

a) Discriminator



✗ Only updates the parameters of the discriminator.

b) Generator



✗ Only one model (Discri. or Generator) is trained at a time.

Noise vector z:

Generated randomly.

Usually by sampling random numbers either between 0 and 1 uniformly or from the normal distribution. ($z \sim N(0,1)$)

Truncation Trick:

✗ Normal dağılımın bazı sınırları içine düşene kadar noise vector (veya başka bir dağılım) z'yi resampling yapar. → Much more realistic result'lar almak için yapılır.

↳ Tabi burada bir trade-off vardır. Nedir? Daha realistic ve high quality image'lar elde etmenize rağmen bu image'lerin diversity'leri daha az olur.

! Truncate etmek demek

↳ Dağılımı sınırlandırmak yani noise'ların mean'e daha yakın olması demek. Bu da kaliteyi arttırır ancak diversity'i azaltır.

Activation functions:

! Non-linear ve Differentiable olmak zorundalar. ✗

✓ ReLU → Suffer from dying ReLU

✓ Leaky ReLU → Solve the dying ReLU problem.

✓ Sigmoid and Tanh → Have vanishing gradient and saturation problems.

Hidden Layer'larda kullanılmazlar genellikle

Batch Normalization:

* Smooths the cost function
* Reduces the covariate shift
* Reduces the internal covariate shift
* Speeds up learning
* During test, the statistics from training are used.
* The batch mean an std. are used for training while the running average statistics from all batches are used for testing.

Padding:

Amaç:

- * Every pixel within the image gets visited the same number of times.
- * Gives similar importance to the edges and the center.

Pooling and Upsampling:

* Pooling reduces the size of input.
* Upsampling increases the size of input.
* Öğrenilen parametreler değillerdir.
  - Upsampling'in öğrenilen parametrelerle yapılmasına Transposed Convolution denir.
    (Transposed Convolution uses a learnable filter to upsample an image)

Common Problems with GAN:

» Binary Cross entropy Loss ile Train edilen modellerin genel problemleri a) Mode Collapse
b) Vanishing Gradients olabilir.

Mode Collapse → Generatör'u eğittik ve yeni resimler generate etmesini bekliyoruz. Generatör her seferinde aynı tip resimler üretiyor ise Mode Collapse problemine yakalanmışızdır. Model eğitilirken Generatör öyle bir data bulur ki Discriminator'u her seferinde kandırabilir. Bu durumda sürekli aynı data ile Discriminator'e gider ve o şekilde eğitilmiş olur. (örn: Discriminatorun 1 rakamını hiç ayırt edememesi ve generatorun bir zamandan sonra sürekli 1 üreterek modeli train etmesi).

BCE Loss in GANs

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\cdot\log h(x^{(i)};\theta) + (1-y^{(i)})\cdot\log(1-h(x^{(i)};\theta))\right]$$

often called as "Minimax" game

Generator } Wants to Maximize that cost
Discriminator } // // Minimize // //

Vanishing Gradient → Discriminator daha iyi oldukça daha az informative geri bildirimler veririz. Gradient'ler 0'a yakınsar ve Generator nasıl gelişeceğini bilemez. (Discriminator'un çok iyi olmasından ötürü ortaya çıkar, Discriminator geliştikçe 0 veya 1'e yakınsayan değerler üretir bu da daha anlamsız geri bildirim olmuş olur Generator için)

BCE'nin karşılaştığı bu "vanishing gradient" problemini "Earth Mover's Distance" Cost function çözer.
↙ (Wasserstein loss bu uzaklığı kullanır.)

Wasserstein Loss

↳ * Approximate the "Earth Mover's Distance".

* All the formulas derive from the "Earth Mover Distance" between the real and generated distributions.

BCE Loss:

* Discriminator outputs between 0 and 1.

W-Loss:

* Critic (~~Discriminator~~) outputs any number. (No longer classifying)
* Try to seperate 2 distributions (fake and real) as much as possible.
* Solves the problem of "Mode Collapse" and "Vanishing Gradient".

Both; Want to maximize the difference between the expected values of the predictions for real and fake.

But;

↳ ✓ Discriminator is bounded between 0 and 1.

✓ Critic is no longer bounded. So it does not classify instances, it outputs number for each instance and "çıktıyı gerçek örnekler için sahte örneklerden daha büyük hale getirmeye çalışır".

✓ Critic Loss: $D(x) - D(G(z))$

↳ Critic tries to maximize it. Yani, it tries to maximize difference between its output on real instances and its output on fake instances.

✓ Generator Loss: $D(G(z))$

↳ Generator tries to maximize it. Yani, it tries to maximize critic's output for its fake instances.

* $D(x)$: Critic's output for a real instance.
* $G(z)$: Generator's output when given noise z.
* $D(G(z))$: Critic's output for a fake instance.

} Output of critic D does not have to be between 0 and 1.

W-Loss'u aşağıdaki şekilde olduğu gibi de ifade edebiliriz:

$$\underbrace{E(C(x))}_{\text{Expected values of the critic's output for the real examples } x.} - \underbrace{E(C(g(z)))}_{\text{Expected values of the critic's output for the fake examples } g(z).}$$

- Generator tries to Minimize this above expression.
- Critic // // Maximize // // //.
- Critic needs to be 1-L Continuous.

Conditional GAN & Controllable Generation

→ Control your GAN.

→ Modify the features in a generated image.

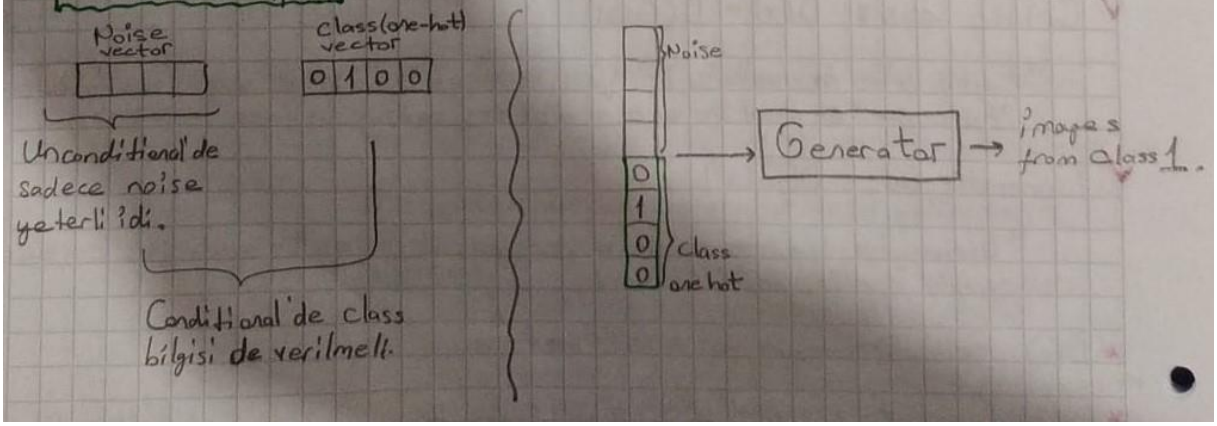
→ Generating examples from determined categories.

1) Conditional GAN:

You can get a random example from the class you specify.

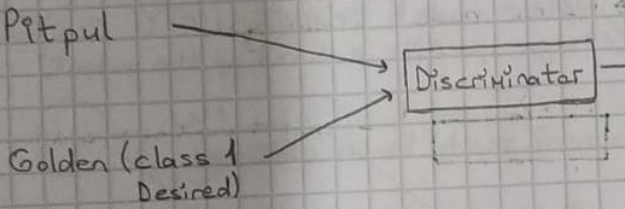
Conditional	Unconditional
* Examples from the classes you want.	* Examples from random classes.
* Training dataset needs to be labeled.	* Training dataset does not need to be labeled.

- Generator Input:



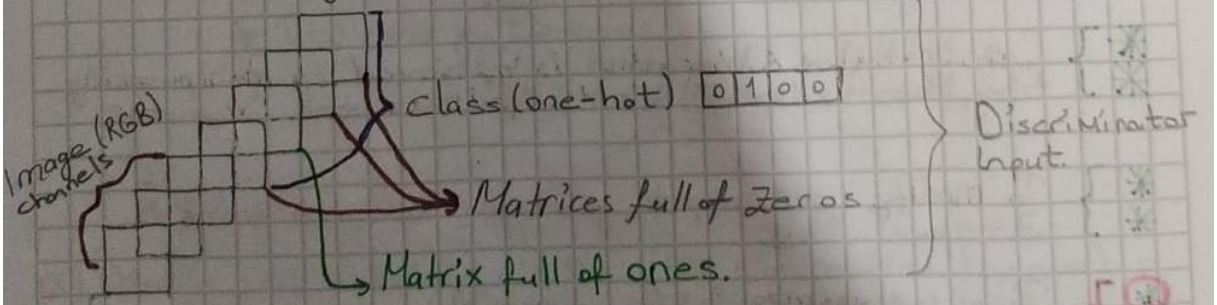
• Discriminator Input:

Generatar'un amacı Discriminator'u fool etmekidi.



Class bilgimiz (Desired) ve generated image (pitpul) aynı anda Discriminator'e verilir. Discriminator mükemmel Pitpullar'ı bile çok düşük oran ile real olarak etiketler. Generate edilen pitpul çok iyi olsa da Class 1 (golden)'a benzemediği için Discriminator bu generated image'a rahatlık ile fake der. Amacımız ne idi? Discriminator'u fool etmek. Bu nedenle Golden (class 1) generated image'leri daha yüksek oran ile real prob. sahip olur ve Discriminator daha fazla fool edilebileceği için hep o class eğitilmeye başlanır.

↳ Class bilgisini image'ye nasıl ekleriz?

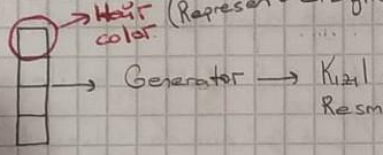


2) Controllable Generation :

GAN tarafından üretilen output'ları kontrol etmenin diğer yolu da "Controllable Generation" yöntemidir.

- * Conditional GAN → Training esnasında label'lar leverage edilir.
- * Controllable Generation → After model training. (Change specific features of the output)

ÖR → Üretilen image'nin mavi saçlı kadın olmasını isteyebiliriz.



Tweak the input Noise vector Z to get different features (mavi saçlı kadın) on the output.

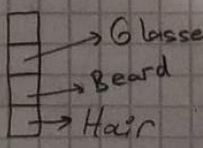
Controllable	Conditional
* Examples with the features that you want.	* Examples from the classes you want
* Training dataset does not need to be labeled.	* Training dataset needs to be labeled.
* Manipulate the Z vector input.	* Append a class vector to the input.

* Controllable Generation is achieved by manipulating the Noise Vector (Z) that is fed into the Generator.

* To modify your output, you move around in the Z-space.

* There are specific indices on the Noise Vectors, these specific dimensions change particular features on the output of your GAN.

Örnek:



"Disentangled Z-space" için geçerli.

Noise

$v_1 = [1, 2, 3, \ldots]$

$v_2 = [5, 6, 7, \ldots]$

z_1 z_2

Hair color — Hair length

"Disentangled Z-space"