Fundamentals of Generative Adversarial Networks

Samia M. BELHADDAD

January 2023

Prerequisites

Basic Calculus, Linear Algebra, Statistics.

Understanding of Deep Learning Models.

Familiarity Classification

Experience with Python and DL frameworks TF - Keras or PyTorch

Contents

1 What are Generative Models?

They try to capture the probability distribution of X, a set of features, knowing or not a Y class.
With added noise, they generate diverse and realistic things, so they try in summary, to make a realistic representation of some noisy input.

Images can be generated, and text, and Numerical data.

Example: A generative model that is trained on a collection of images to be able to generate similar images.

The most popular Generative Models are two:

1.1 Variational AutoEncoders VAEs

VAEs work with two models which are neural Networks:

They learn by feeding realistic images to the encoder, then the encoder learns the best way to represent that image in a latent space.

VAE takes a latent representation and puts it into a decoder. Decoder: tries to reconstruct the representation that was an output of the encoder.

The encoder encodes the image into a distribution from which values are sampled to be fed into the decoder.

After training, we log off the encoder, pick latent points from latent space and decode them into new images. This is done through the decoder which would have learned how to decode well.

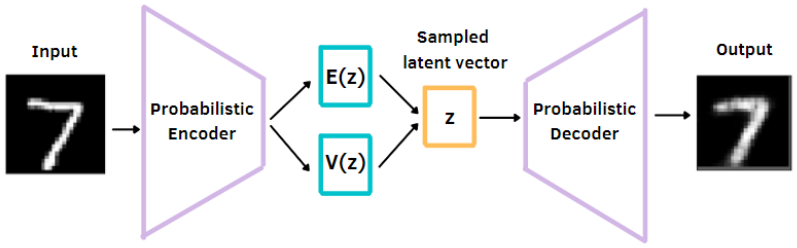


Figure 1: VAE Architecture Example

1.2 Generative Adversarial Networks GANs

GANs Learn to produce realistic objects that are hard to distinguish from real existing ones. They are composed of two models which learn by competing with each other Generator and Discriminator.

Generator takes as input some random noise and produces a fake output (image for example).

Discriminator learns to differentiate between real data and fake ones.

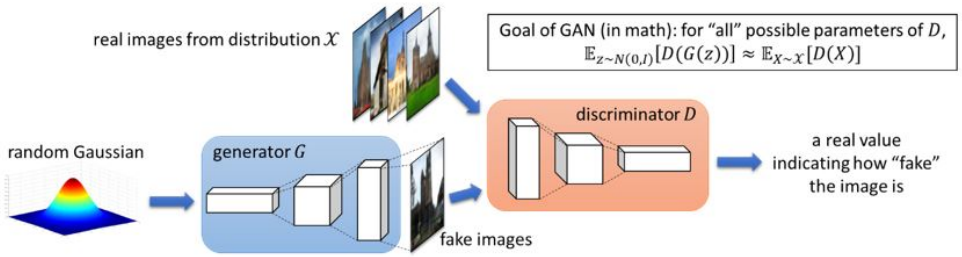


Figure 2: GAN Architecture Example

1.3 Discriminative Models

Discriminative Models are a type of ML model that learns how to distinguish between groups or classes.
They take a set of features and they learn or determine categories.
They model the probability : P(Y|X)

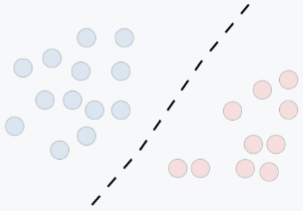
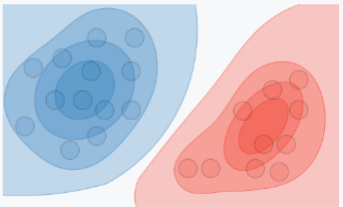
	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Figure 3: Discriminative VS Generative Model

1.4 The intuition behind GANs

1.4.1 Discriminator

Learns to distinguish real from fake objects.

A discriminator is a Neural Network. It is a type of classifier that differentiates between different classes.

Classifiers take a set of features X, predict a class for the features, compare the predicted class with the real class already known, and then update their parameters according to the gradient of the loss function.

Discriminators model : P(Y|X) = P(Class| Features)

1.4.2 Generator

Learn to produce fake objects that look real to fool the discriminator. It is a neural network that takes a noise vector which is a distribution as input and outputs features of an object (eg: image).

The generator learns by producing objects that the discriminator identifies are real or fake.

It wants its generated objects to be as real as possible.

Generators model P(X | Y) =P(Features | Class) or P(X) =P(Features)

1.4.3 Generator and Discriminator being Adversaries

The Discriminator will have access to a set of images, some of them are real paintings, some are not, at first it won't know which one is real and which one is not, within training phases, it will be told which is real and which is fake (while comparing real vs predicted classed) and that will make it know whether they are real or fake.

The Generator, while producing its forged painting, will know what direction to go by looking at the scores assigned to its work by the discriminator.

Both will improve over time.

The competition ends based on where one wants it to stop.

Once a generator generates well, its parameters can be saved and frozen Samples could be sampled from this generator using the optimal parameters.

Example: Art painting generation

The generator will take an input which in general is modeled as noise to produce a painting.

The discriminator will compare the painting with real images of the painting he has access to, to know whether what the generator made was a real painting or not.

1.4.4 GAN Architecture

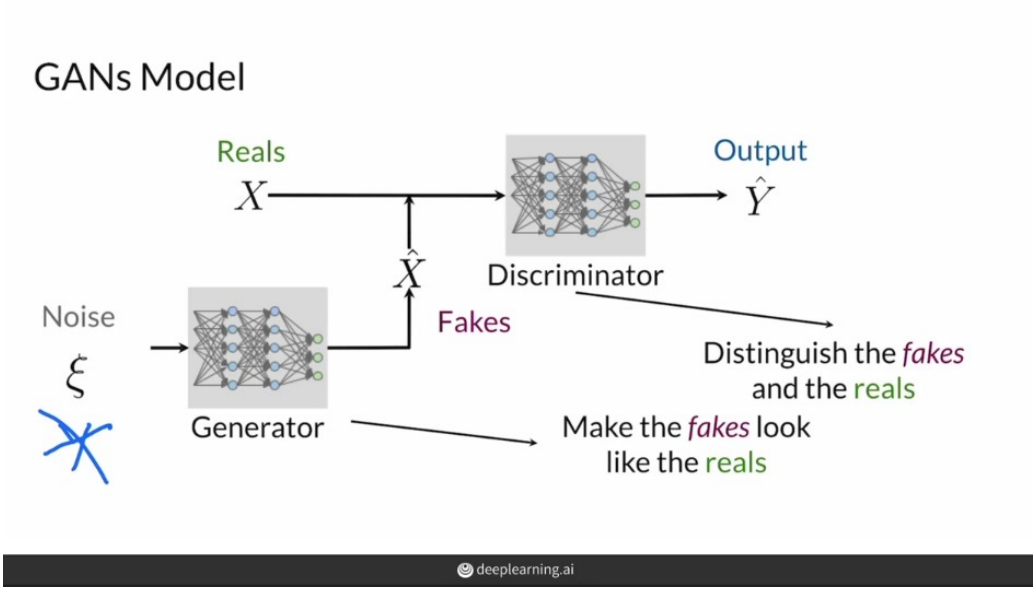


Figure 4: Generative Adversarial Network Model

A noise is passed into a Generator.

The generated examples and real examples are passed to the discriminator.

The Goal of the discriminator is to distinguish between the generated examples and the fake examples.

The Goal of the Generator is to fool the discriminator by generating examples that look as real as possible.

2 Training Generative Adversarial Networks

The training of a generator and discriminator are alternated.

Both models should improve together and should be kept at a similar skill level. If u had a really skilled discriminator, it will be telling u that everything is fake, and doesn't help the generator learn.

If the Generator is super skilled; generated images will always seem too real to the discriminator.

Figure 5: Case Generator is SuperSkilled

2.1 Loss Function: Binary Cross Entropy

BCE is useful for gans because it's a suitable design for binary classification of the two categories: real and fake.

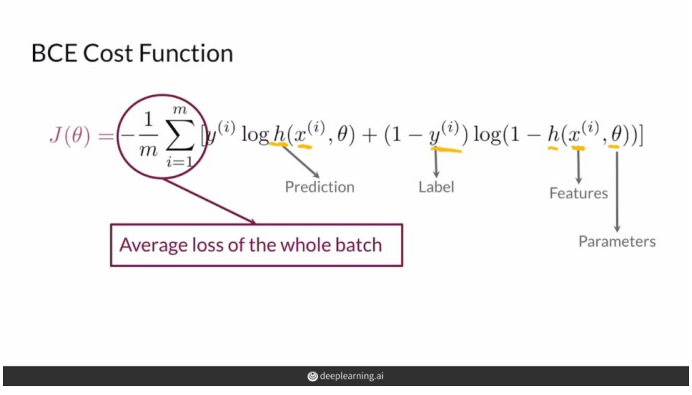


Figure 6: Binary Cross Entropy Formula

The first term of the BCE is only relevant when the label is 1.

If the label (Yi=0) the prediction is any (0 or 1), it gives 0 anyway.

If label (yi=1) the prediction matters:
Either y pred ∼0.9 or y pred = 0.
it gives values ∼ 0 or -inf respectively.

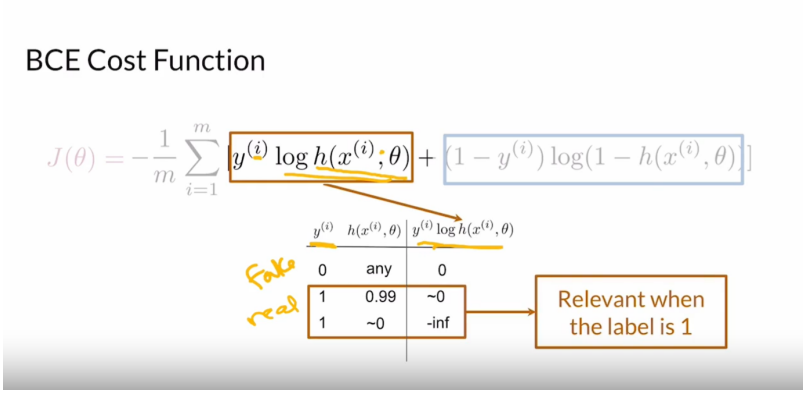


Figure 7: Left term of BCE

The second term of the BCE is only relevant when the label is 0.
If the prediction is really good(fake and (∼0.01) it evaluates close to 0.

If the prediction is terrible (close to 1 it evaluates to -inf respectively.

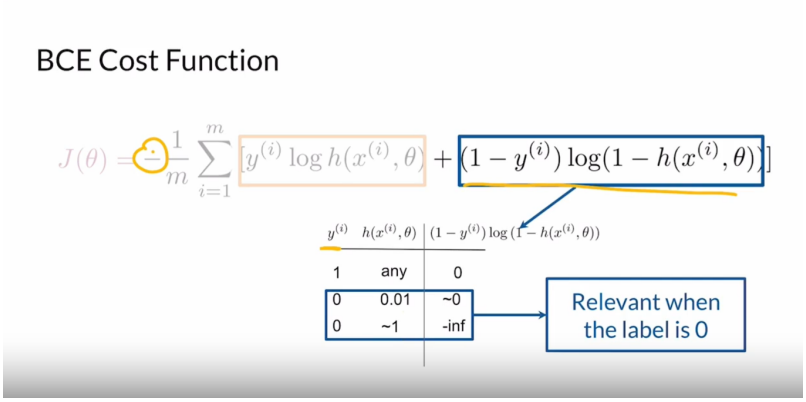


Figure 8: Right term of BCE

2.2 Discriminator Training

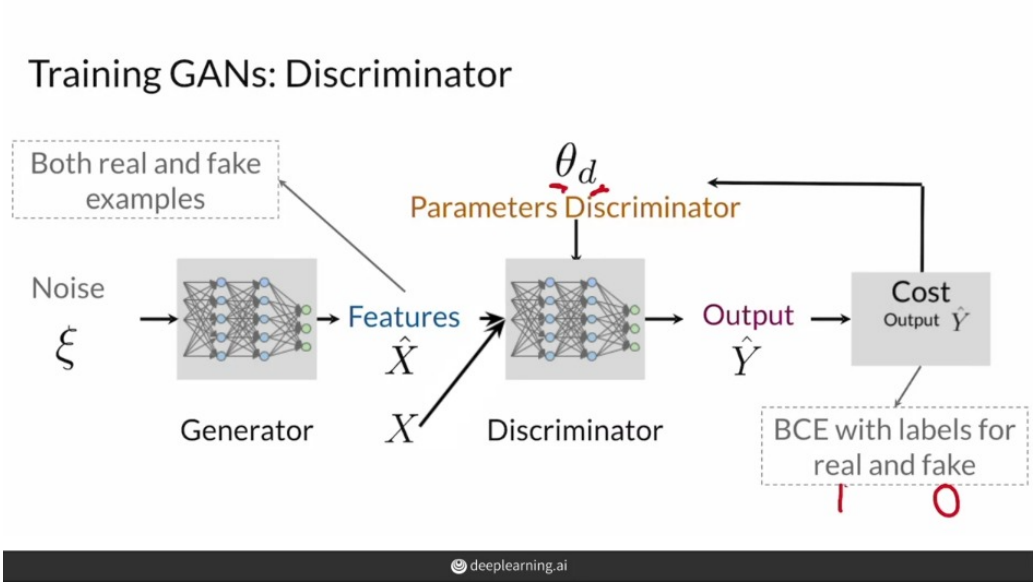


Figure 9: Discriminator Training

The discriminator gets fakes generated by the generator from input noise and real images.

The Discriminator doesn't know which of the examples is real or fake.

Discriminator makes predictions, it gives a score of how fake or how real are input examples.

Predictions are compared with desired true labels (fake or real) using BCE Loss.

The parameters of discriminators are updated.

This updates the parameters of the discriminator only.

2.3 Generator Training

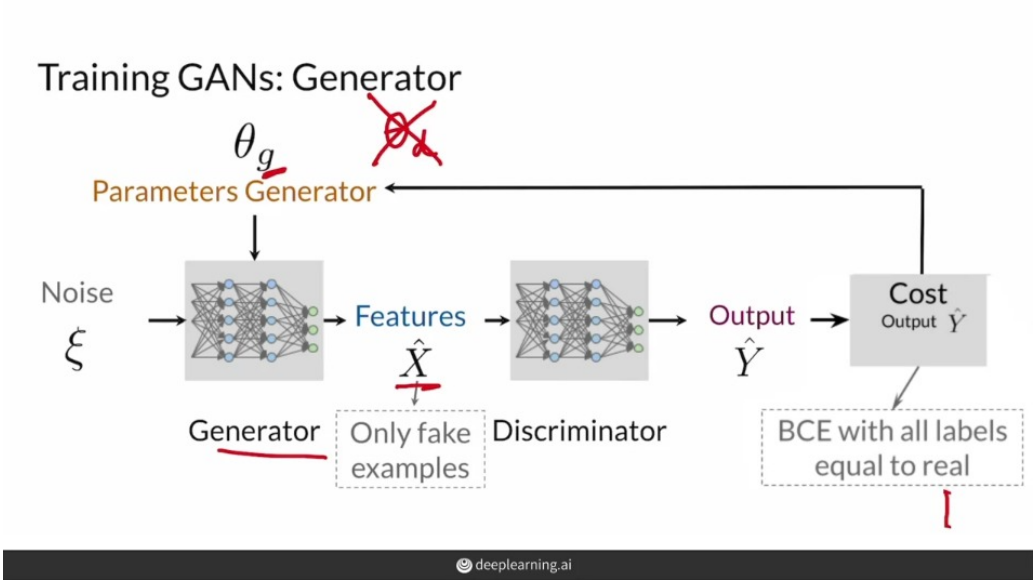


Figure 10: Generator Training

It learns to generate a few fake examples from noise.

The generator only sees fake examples that it outputs.

Discriminator makes predictions of how real or fake they are, predictions are compared using BCE with all labels equal to real.

Discriminator wants fake examples to seem as fake as possible The generator wants fake images to look as real as possible.

After computing the cost,the gradient is propagated backward and the parameters of the generator are updated.

3 GANs Architecture Implementation

3.1 GANs Architecture in Tensorflow

Define Generator as a Neural network with 3 dense layers.

```
Generator = keras.models.Sequential(
  [
    keras.layers.Dense(100,activation = "selu",input_shape = [size]),
    keras.layers.Dense(150, activation = "selu"),
    keras.layers.Dense(28*28, activation ="sigmoid"),
    keras.layers.Reshape([28,28])
    ])
```

Define Discriminator as a binary Classifier .

```
Discriminator = keras.models.Sequential(
  [
    keras.layers.Flatten(input_shape = [28,28]),
    keras.layers.Dense(150,activation = "selu"),
    keras.layers.Dense(100, activation = "selu"),
    keras.layers.Dense(1, activation ="sigmoid"),
  ])
```

Define full GAN Model.

```
GAN = keras.models.Sequential([Generator,Discriminator])
```

Compile GAN and Discriminator .
The generator is not compiled , it is trained through GAN model.
Discriminator must be untrained when GAN is trained .

```
Discriminator.compile(loss = "binary_crossentropy", optimizer = "rmsprop")

Discriminator.trainable = False

GAN.compile(loss = "binary_crossentropy", optimizer = "rmsprop")
```

Next Step : Load data and Define a Training Loop

3.2 GANs Architecture in Pytoch : OOP

Define Generator Class

```
class Generator(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = nn.Sequential(
                nn.Linear(100, 10000),
                nn.LeakyReLU(),
                nn.Linear(10000, 4000),
                nn.LeakyReLU(),
                nn.Linear(4000, flat_image_size),
       )


    def forward(self, latent_space):
        latent_space = latent_space.view(1, -1)
        out = self.linear(latent_space)

        return out
```

Define Discriminator Class

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = nn.Sequential(
            nn.Linear(flat_image_size, 10000),
            nn.ReLU(),
            nn.Linear(10000,1),
            nn.Sigmoid()
        )

    def forward(self,img) :
        img = img.view(1, -1)
        out = self.linear(img)

        return out
```

Next Step : Compile , Load data and Define a Training Loop

4 Resources

Book : Hands on Machine Learning with ScikitLearn,Keras and TensorFlow.

https://www.coursera.org/specializations/generative-adversarial-networks-gans

https://developers.google.com/machine-learning/gan/generative

https://www.tensorflow.org/tutorials/generative

https://towardsdatascience.com/tagged/generative-model

https://www.microsoft.com/en-us/research/blog/how-can-generative-adversarial-networks-learn-real-life-distributions-easily/