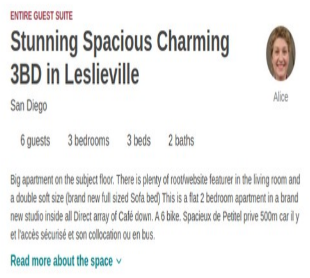
Generative Adverasrial Networks (GANS)

Learning Objective :-

- ✔ What does it mean by Generative Adversarial Networks ?
- ✔ What does it mean by Auto Encoders ?
- ✔ Why do i need to study Generative Adversarial Networks ?
- ✔ Why do i need to study Auto Encoders ?
- ✔ What is the history of Generative Adversarial Networks ?
- ✔ What is the history of Auto Encoders ?
- ✔ Who created Generative Adversarial Networks and why is it now viral ?
- ✔ Who created Auto Encoders
- ✔ What does the name Generative Adversarial Networks refers to ?
- ✔ What does the name Auto Encoders refers to ?
- ✔ How to implement Generative Adversarial Networks ?
- ✔ How to implement Auto Encoders ?

Generative Adversarial Network comes from the concept called Generative models , which is a process of randomly generating new data that looks very similar to the training data. Faces generated by Generative Adversarial Networks are now hard to believe that the people they represent do not exist. we can judge ourselves by visiting websites like https://thispersondoesnotexist.com/ a website that shows faces generated by a recent GAN architecture called Style GAN. The GAN used in was trained on a large data set of human faces, and it outputs a plausible picture of a human face not in the training set.

We can also check https://thisrentaldoesnotexist.com/ to see some generated Airbnb bedrooms. What is Airbnb ? Airbnb is a community built on a sharing economy. It is an online platform that allows property owners to list their place as holiday accommodation and allows travelers to find a place to stay while they are away from home.



ENTIRE GUEST SUITE

Stunning Spacious Charming 3BD in Leslieville

San Diego

Alice

6 guests 3 bedrooms 3 beds 2 baths

Big apartment on the subject floor. There is plenty of root/website featurer in the living room and a double soft size (brand new full sized Sofa bed) This is a flat 2 bedroom apartment in a brand new studio inside all Direct array of Café down. A 6 bike. Spacieux de Petitel prive 500m car il y et l'accès sécurisé et son collocation ou en bus.

Read more about the space ⌄

Why do we need to study Generative Adversarial Networks ?

- Generative Adversarial Networks used for generating new faces or new data that has never existed. So on the supervised learning side , if we want to classify scratched objects (Visual Inspection) or classify a medical x-ray images but you do not have enough pictures , so we will use GANS to synthesize(generate) more data to feed to our learning algorithm.

- We can use Generative Adversarial Networks for super resolution of an image , it takes a low resolution image and then turn it in to a high resolution image. How ?

- We can use Generative Adversarial Networks in some of our funny applications to make some one look younger or older , how ? and what is the point of doing these , philosophically ?

Generative Adversarial Networks used for deep fake creation. Deep Fake content is created by using two algorithms that compete with one another. One is called a generator and the other one is called a discriminator. The generator creates the fake digital content and asks the discriminator to find out if the content is real or artificial.

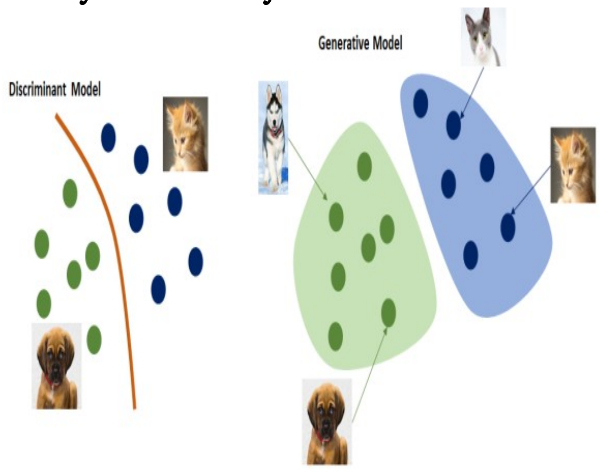
Generative and Discriminative Models :-

The most important difference between naive Bayes and logistic regression is that logistic regression is a discriminative classifier while naive Bayes is a generative classifier. These are two very different frameworks for how to build a machine learning model.

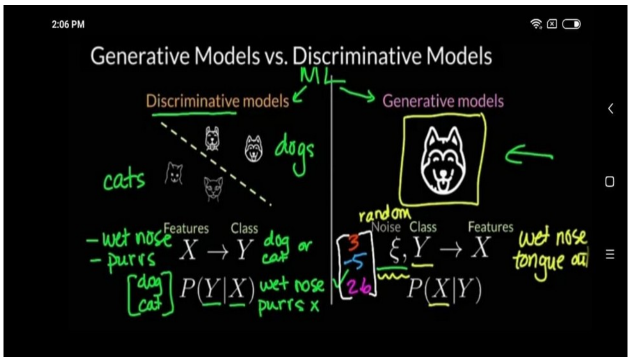
Story:

A father has two kids, Kid A and Kid B. Kid A has a special character whereas he can learn everything in depth. Kid B have a special character whereas he can only learn the differences between what he saw. One fine day, The father takes two of his kids (Kid A and Kid B) to a zoo. This zoo is a very small one and has only two kinds of animals say a lion and an elephant. After they came out of the zoo, the father showed them an animal and asked both of them “is this animal a lion or an elephant?” The Kid A, the kid suddenly draw the image of lion and elephant in a piece of paper based on what he saw inside the zoo. He compared both the images with the animal standing before and answered based on the closest match of image & animal, he answered: “The animal is Lion”. The Kid B knows only the differences, based on different properties learned, he answered: “The animal is a Lion”. Here, we can see both of them is finding the kind of animal, but the way of learning and the way of finding answer is entirely different. In Machine Learning, We generally call Kid A as a Generative Model & Kid B as a Discriminative Model.

Consider a visual metaphor: imagine we're trying to distinguish dog images from cat images. A generative model would have the goal of understanding what dogs look like and what cats look like. You might literally ask such a model to 'generate', i.e., draw, a dog. Given a test image, the system then asks whether it's the cat model or the dog model that better fits (is less surprised by) the image, and chooses that as its label. A discriminative model , by contrast, is only trying to learn to distinguish the classes (perhaps without learning much about them). So maybe all the dogs in the training data are wearing collars and the cats aren't. If that one feature neatly separates the classes, the model is satisfied. If you ask such a model what it knows about cats all it can say is that they don't wear collars.



Generative model expresses how to generate the features of a document if we knew it was of class c , which means P(X|Y), the probability distribution of x the different features given the class y and the random noise data , why do i need the random noise ?. By contrast a discriminative model in this text categorization scenario attempts discriminative model to directly compute P(Y|X) , the probability of being in that class given the feature X. Perhaps it will learn to assign a high weight to document features that directly improve its ability to discriminate



An Art Forger and An Art Inspector, Intuition Behind GANS

There are two ways to look at a GAN.

Call it an artist that sketches realistic images from scratch. And like many successful artists, it too feels the need of a mentor to reach higher levels of proficiency. Seen thus, a GAN consists of:

- An artist, that is , the Generator

- And a mentor, that is , the Discriminator

The Discriminator helps the Generator in generating realistic images from what is merely noise.

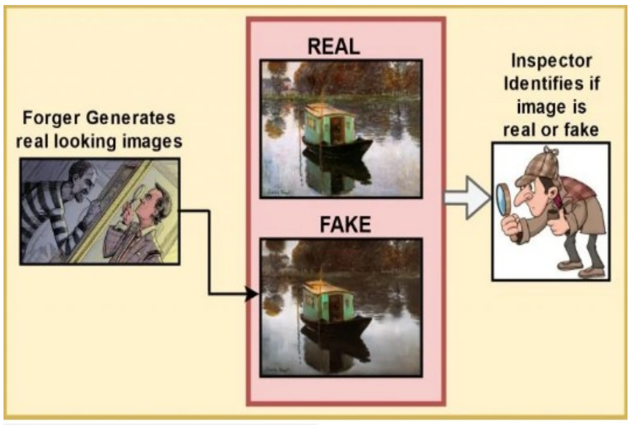

Forger Generates real looking images
REAL
FAKE
Inspector Identifies if image is real or fake

Figure :- The concept of GANS through an art forger and an art inspector.

What if the GAN was not really an artist, but an 'art forger'. Wouldn't it need an inspector then to check what is genuine and what is not ? Look at a GAN this way, then:

- The generator plays the role of the art forger. The aim of this network is to mimic realistic art.

-While the discriminator inspects, whether the art is real or fake. It's job is to look at the real as well as the fake artwork generated by the forger, and to differentiate between the two. Further, the art inspector employs a feedback mechanism to help the forger generate more realistic images.

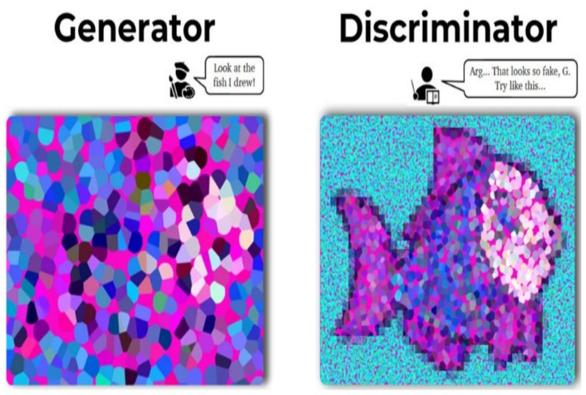


Figure 5. The forger trying to generate fake images using a feedback mechanism from the inspector.

In short , as shown above, GAN is a fight between two nemeses: the generator and the discriminator.

The generator tries to learn the data distribution, by taking random noise as input , and producing realistic-looking images.

On the other hand, the discriminator tries to classify whether the sample has come from the real data set, or is fake (generated by the generator).

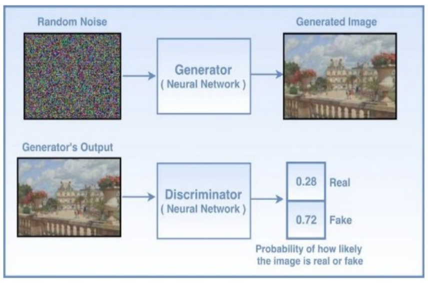
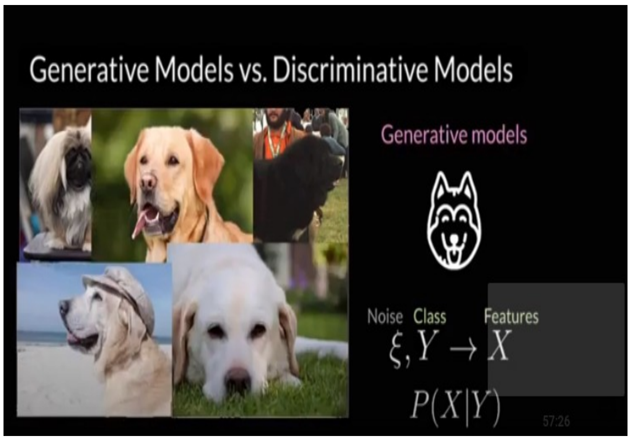


Figure 6. The generator tries to generate fake images while taking random noise as input and the discriminator tries to classify it as real or fake.

Question :- Why is the input data a random noise data ? If we want to generate the image of a dog , why not give him training examples of a dog and ask our generative model , " Hey generate me a dog ? "

The noise in here is to ensure that what's generated isn't actually the same dog each time because generating just one dog is no fun and also pointless.



In good run of a Generative model we could get a picture of this pecking geese and in another run a Tibetan mastiff and if we could continue to run it multiple times without any restrictions we will end up getting more pictures representing the data set our generative model was trained on.

Up to when should we need the feed back mechanism of our discriminative model to the generative model ?

The generator takes in some kind of random noise input and as an output it can generate the dog overtime of course, in the beginning it may have some evil eyes , then the discriminator looking up fake and real images and simultaneously trying to figure out which ones are real and which ones are fake , over time each model tries to one up each other , so these two models will compete against each other which is why they are called Adversarial. Adversarial means :- involving two people or two sides who oppose each other. So we can imagine those muscles growing over time and learn from each other until they reach a point where we don't need the second model any more the discriminator and the generator can take in any random noise and produce a realistic image.

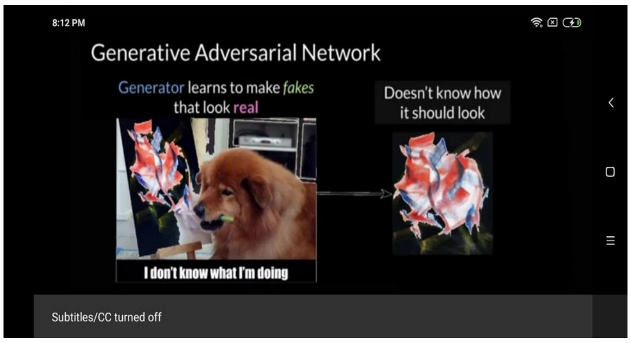
There are many types of Generative models , for now we will briefly introduce to the most popular ones.

Generative Adversarial Learning

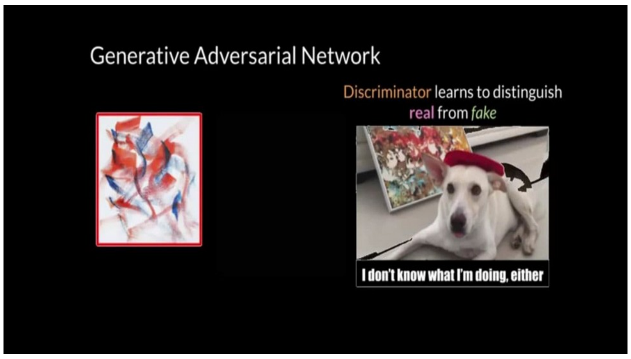
The Generative and the Discriminative models are typically two different neural networks. The generator learns to generate fakes that look real to fool the discriminator and the discriminator learns to distinguish between what's real and what's fake. So we can think Generator as a painting forger and we can think the Discriminator as an Art inspector.

In order to fool the discriminator the generator will try to forge paintings that look more like the real ones and in order to catch the generator the discriminator will try to learn how to not get fooled even by the closest replica.

At the beginning of this game the generator actually isn't that much sophisticated it doesn't know how to produce real looking art work, Additionally the Generator is not allowed to see the real images , it does not know how this painting should even look.


8:12 PM
Generative Adversarial Network
Generator learns to make fakes that look real
Doesn't know how it should look
I don't know what I'm doing
Subtitles/CC turned off

For the discriminator at the beginning doesn't know what is real and what's fake but on the discriminator case it's allowed to look the real art work it's just jumbled up with the fake ones as well and the model doesn't know which one is which that is for the model to figure out and learn how to decide.


Generative Adversarial Network
Discriminator learns to distinguish real from fake
I don't know what I'm doing, either

Questions :-

- How the Discriminative model works

- How the Generative model works

- If the Generative model can not see the real image examples , how does it learn to generate the realistic examples , how does the information of generating came from?

- Why is not allowed for the Generative model to not access the real images ?

- How the Generative Model improves over time ?

- How the feed back mechanism of our discriminator works

- Who sets the noise vector or how to set the noise vector and what is this noise vector

- How do i know when i exactly need to stop the discriminative model giving feedback to the generative model

- How to calculate the loss function of the discriminative model ?

- How to update the parameters of the generative model after we are calculating the loss using the discriminative model

Application Of GANS

GANS For Image Translation:- Which means they can take image from one domain and transform it in to another domain. For instance we can transform the image of a horse to a zebra and vice versa , what is really interesting is we don't actually need a zebra and a horse doing the same things but instead just transfer that style over , for this we used Algorithm called Cycle GANS.

https://www.youtube.com/watch?v=pyUFYpietYw

GANS can also take a still life portrait for example monalisa and animate it using the motion of any real person's face so they don't really need to look like monalisa to play the part.

https://www.youtube.com/watch?v=GnCDJ5qQ1Lo

Companies that are using GANS

- Google is using GANS for text Generation
- IBM is using GANS for Data Augmentation
- Snap chats and Tiktok used for Image filters
- Disney using for Image Resolution

Discriminative And Generative Models

Discriminative Models :- Typically used for classification in machine learning. Learn how to distinguish between classes such as Dogs and Cats and often called Discriminative Models.

Discriminative Models take a set of features X such as having a wet nose or a purse and from these features determine a category y which is whether is a Dog or a Cat.

Features	Class
X --------->	y

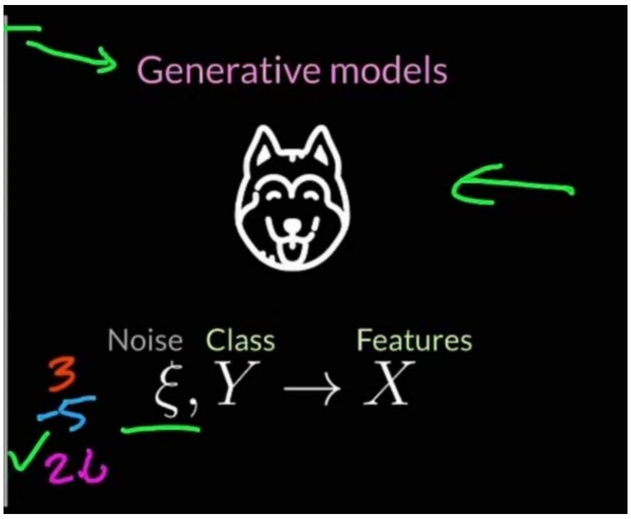
In other words Discriminative models try to model the probability of a class Y given a set of features X. p(y|x)

Generative Models :- On the other hand Generative models try to generate some realistic representation of some class , for instance a realistic representation of a dog.

They take some random input represented by the noise , which could take on the value , let's say [3 , -5 , 2.6]. The point is the noise represents a random set of values going in to the generative model.

The generative model takes a class Y (such as a dog) and from this inputs , it generates a set of features X , that look like a realistic dog.

Why we need the noise in the first place ?



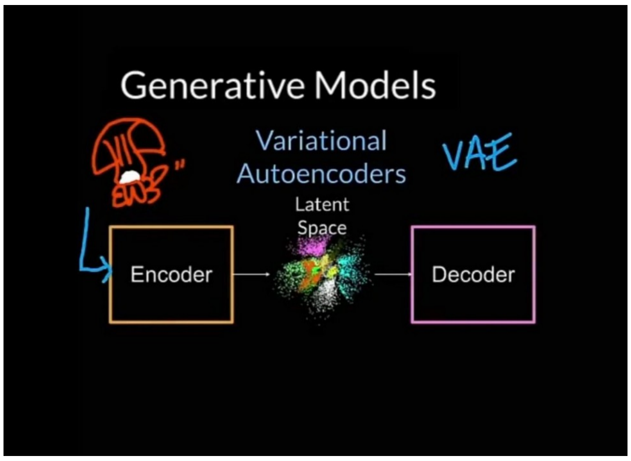
Why can't we just tell it , hey generate me a dog for me and it will generate a dog but the noise in here ensure that what's generated isn't actually the same dog each time. Generating just one dog is no fun and pointless.

Generative model try to capture the probability of X , the different features of having a wet nose , a pointy ears given the class y of the dog and with added noise these models will generate diverse representations of the class y. Mathematically this means the probability of X given y :- P(X|y).

Types of Generative Models

1) Variational Auto Encoder (VAE)
2) Generative Adversarial Networks

1) Variational Auto Encoder (VAE) :-

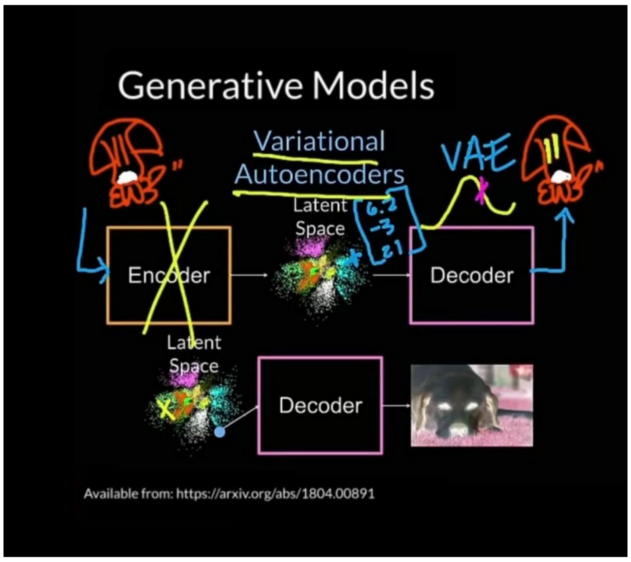


Variational Auto Encoders work with two models an encoder and decoder and these are typically neural networks.

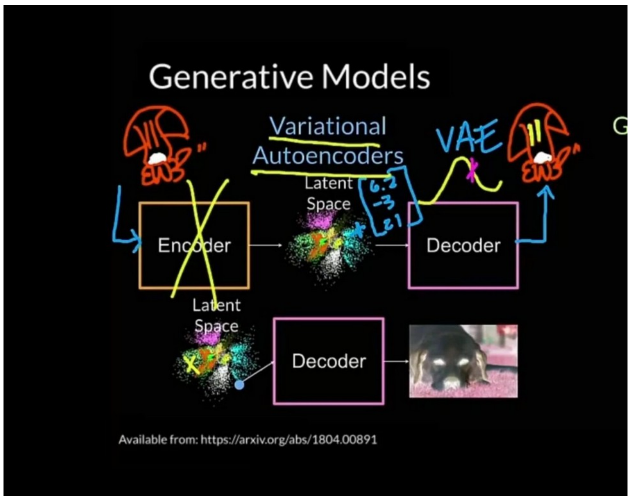
They learn by first feeding realistic images in to the encoder , such as a realistic image of a dog then the encoder's job is to find a good way of representing that image in to this latent space. Latent spaces are vector spaces or embedding vectors that encodes a meaningful internal representation of the features.

Let's say our realistic image is represented in to the latent space by the vector of numbers like [6.2 , -3 , 21] and what the VAE does now is take this latent representation and put it in to the decoder and the goal of the decoder is to reconstruct the realistic image that it has seen before.

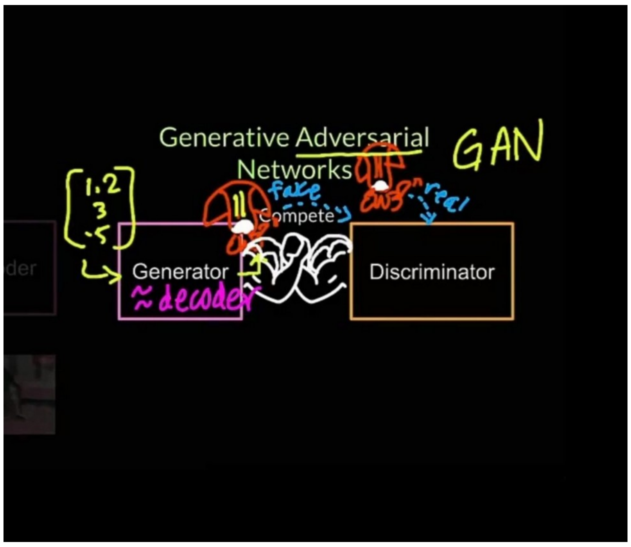
In the beginning the decoder won't able to be able to reconstruct the image and may be the dog produced will have evil eyes. Now after a well training we actually turn off the encoder and we can pick random points in the latent space and the decoder will have learned to produce realistic image of a dog.



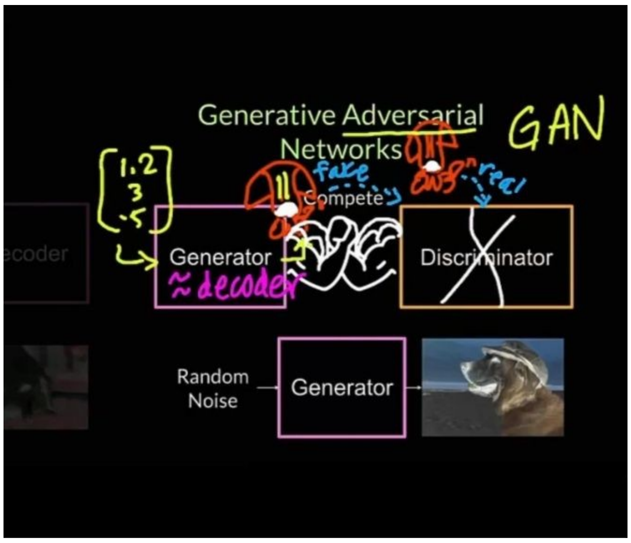
Until now we only described is largely the Auto Encoder part , the variational part actually injects some noise in to the model in training process and instead of having the encoder encode the image in to a single point in the latent space , the encoder actually encodes the image on to a whole distribution , what's that actually mean ? and how do we encode the image on a whole distribution. and then samples a point on that distribution to feed in to the decoder to then produce a realistic image , this adds a little bit of noise because since different points can be sampled on this distribution.



2) Generative Adversarial Networks



GANs are composed of two models but now there's a Generator which Generates image like the Decoder. The Generator takes in some kind of random noise input for example [1.2 , 3 , 5] as a vector and that is input to the generator and of course an optional class of a dog (y) but if we're generating just dogs , we don't need to input that as output it can generate that same dog , overtime of course , which is in the beginning it might look an evil , so the generators role in some sense is very similar to the decoder in the VAE so what's different is that there's no guiding encoder but instead there is a discriminator looking at fake and real images and simultaneously trying to figure out which ones are real and which ones are fake , over time each model tries to one up each other , so these models compete against each other which is why they're called adversarial. So they will continue to learn to each other and compete to each other until to reach to a point where again don't need the second model anymore , the discriminator and the Generator will take any random noise and produce a realistic image.



Summary :-

- ✔ Generative Adversarial models learn to produce realistic examples , kind of like an artist that can paint that look like photos.
- ✔ Mean while the discriminator models distinguish between different classes
- ✔ A discriminative model can be a sub component of a generative model such as the discriminator whose classes are real and fake.

Intuition Behind GANS

We call that GANS have two components , one is the Generator and the other is called the discriminator and these are typically two different neural networks. The Generator learns to generate fakes that look real to fool the discriminator and the discriminator learns to distinguish between what's real and what's fake. So we can think of the Generator as the paining forger and the discriminator as an art inspector , so the Generator forges fake images to try to look as realistic as possible and it does this in the hope of the fooling the discriminator.

The discriminator here have both real famous painting and a fake ones created by the generator and tries to tell which ones are real and which ones are fake.

So in order to fool the discriminator the Generator will try to forge paintings that look more like the real ones and in order to catch the generator the discriminator will try to learn how to get not fooled even by the closet replica. So to start this game all we need is a collection of these real images , like famous paintings.

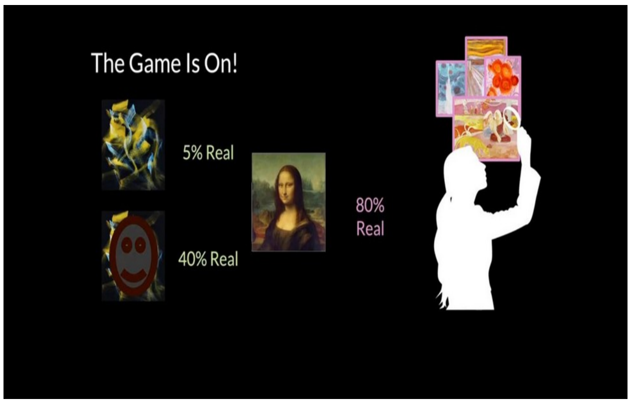
So at the beginning of these Game , the Generator actually isn't sophisticated , it doesn't know how to produce real looking artwork , additionally the Generator isn't allowed to see the real images , it doesn't know how this painting should even look. so at the very beginning the elementary generator initially just paints something with evil eyes because nobody tells the Generator what to do and what to Generate.

The other beginning component is actually an elementary discriminator that doesn't know what's real and what's fake , but in these case the discriminator allowed to look the real artwork which is jumbled up with the fake ones as well and the discriminator doesn't know which one is which and it's for the discriminator to figure out and learn how to decide.

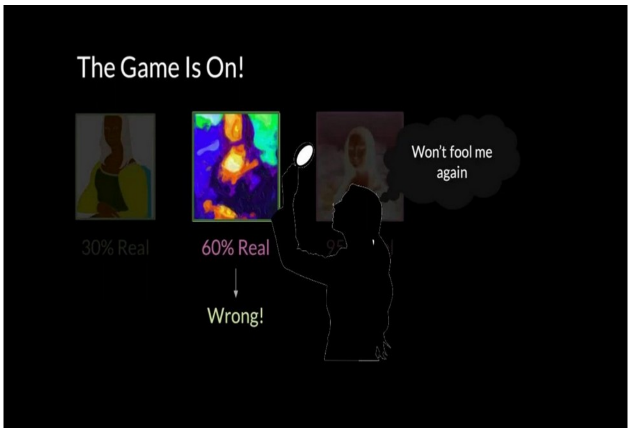
So to start the competition we train the discriminator using the real artwork so it's able to know which images are actually real , so after it decides "may be this looks real " so after it decides “may be this looks real” we (Comparing the prediction with the real label) actually tell it yes that’s real or no that’s fake , this way we can get the discriminator that’s able to differentiate poorly drawing image from the one’s that are slightly better.

The discriminator gets these all jumbled up so it doesn’t know up front which ones are real and which ones are fake but we tell it as it learns which ones is real and which ones are fake , whether it’s right and wrong in determining these two classes.

When the generator produces a batch of paintings the generator will know in what direction to go on and improve by looking at the scores assigned to the Generator’s work by the discriminator.



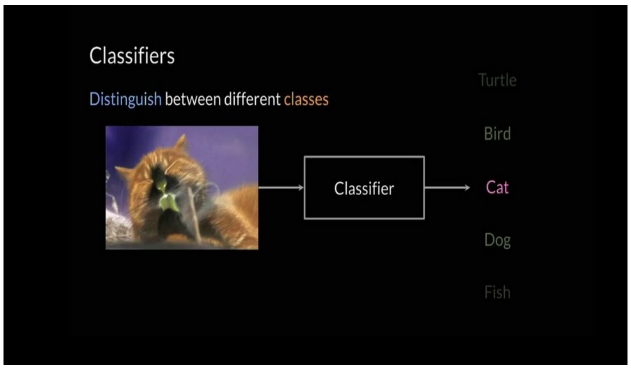
In discriminator also improves over time because it receives more and more realistic images at each round from the Generator and always remember that the discriminator receives both real and fake images all jumbled up in a pile but essentially it tries to develop a keener eye as the images get better.



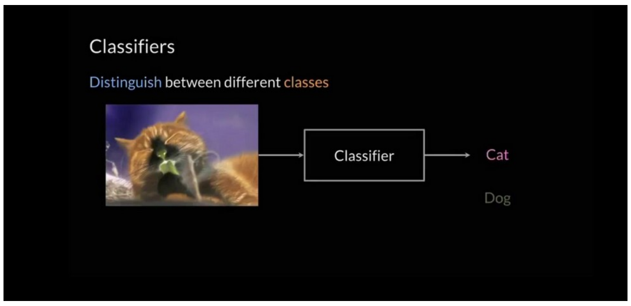
So when the discriminator decides that this image 60% real created by the generator is 60% real but we actually tell it after it says 60% real , that its wrong , that’s not necessarily real , that’s actually fake and after many rounds , the generator will start producing painting that are harder and harder to distinguish , if not possible for the discriminator to distinguish from the real one and at this point the person who wants a good generator to generate awesome fake images , when we are happy with the results of this Generator the game will end. (When it says the person , it wants to refer the loss function which consistently evaluates the prediction of the discriminator) and then it reaches the kind of accuracy we want we will stop and the game will end.

How the Discriminator Really Works

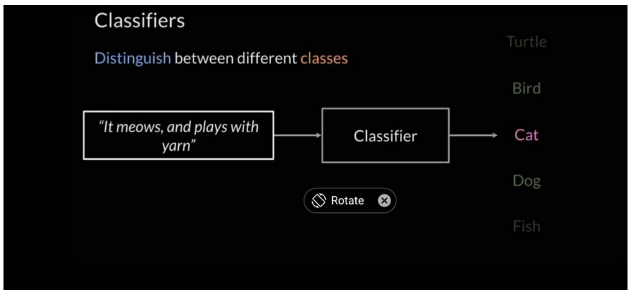
The goal of a classifier is to distinguish between different classes , given a certain image of a cat the classifier should be able to tell it’s a cat or not a dog.



In fact it can learn to differentiate cats from multiple different classes and this depends on which classes we want to differentiate.

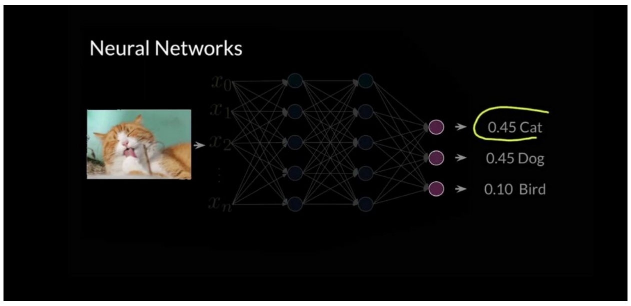


Classifiers are not limited to determining Image classes so we could have this piece of text and will classify that it's a cat or a do.


Classifiers
Distinguish between different classes
"It meows, and plays with yarn"
Classifier
Turtle
Bird
Cat
Dog
Fish
Rotate

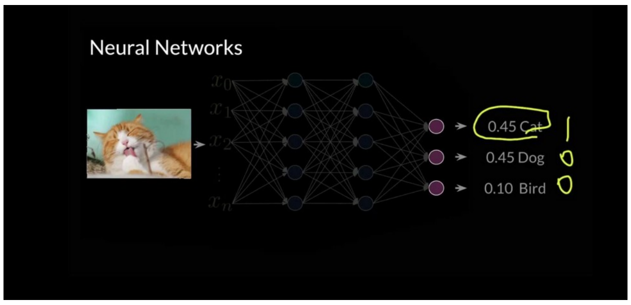
We could have a video chat of a cat.

One type of model for a classifier is using a neural network and this neural network will take some feature x (the n-different features) . it computes the non-linearities and outputs the probability for a set of categories.



In the beginning of the model , it will probably won't know how to classify correctly but it will learn over time trying to improve its predictions according to the true labels from the data.

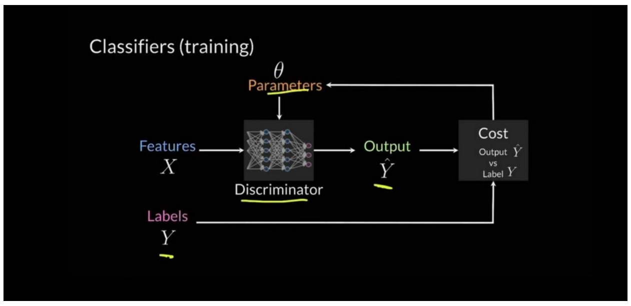
So at the end we will penalize for the incorrect predictions by calculating the loss in accordance with the correct label.



So we will repeat this process until our classifier is in a good shape. The goal of the discriminator is to model the probability of each class given a set of input features.

P(cat | features of the cat) = p(y_class|X_features)

So the discriminator is a classifier that inspects the fake and the real examples and determines whether they belong to the real or fake class.

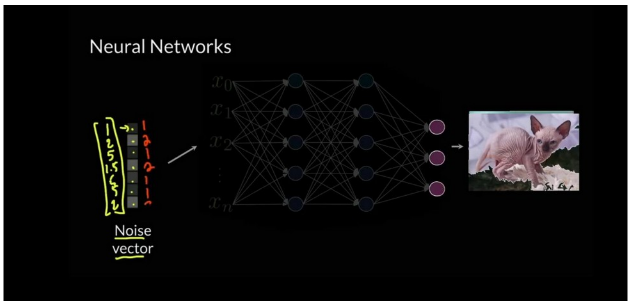


P(Fake|features of monalisa) = 0.85 -> Fake

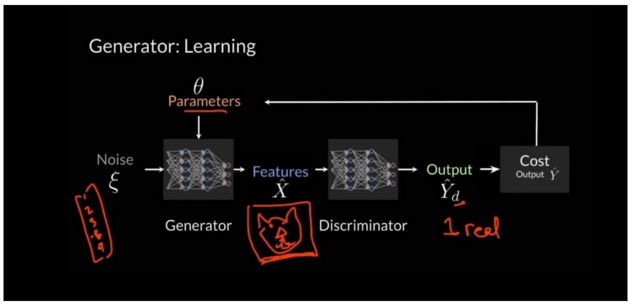
Generator :-

The Generator's final goal is to be able to produce examples from a certain class , if we trained it from the class of a cat then the generator will do some computations and output a representation of a cat that looks real.

The Generator won't output the same cat at every run and so to ensure it's able to produce different examples every single time we actually will input different set of random values , also known as the a noise.



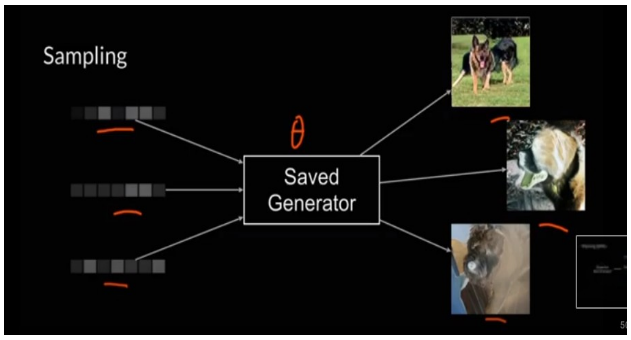
different shaded cells of the noise are just different vector values. So this features starting from X1 to Xn includes the class and the numbers in this noise vector. So this noise will be fed in to the input with that class y for cat in to the Generator Neural network. So then the neural network will compute a series of non - linearities from those inputs and return some variables that look like a cat. So instead of different classes its output will actually be an image , may be an image that has three million pixels , so the output will have three million nodes at the end and in another run it will generate another bread of a cat.



The Generator wants the prediction to be as close to 1 , meaning real as possible where as the discriminator is trying to get the prediction to be zero , which is fake.

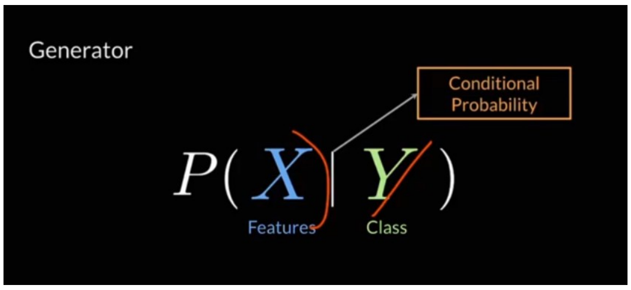
Then after computing the cost , it then update the parameters of the generator and that gets to improve over time and know which direction to move its parameters to generate something that looks more real and will fool the discriminator.

So once we get a generator that looks pretty good we can save the parameters theta of the generator and that typically means freezing those theta values and saving it some where and we can load it back up and then sample from this saved generator.

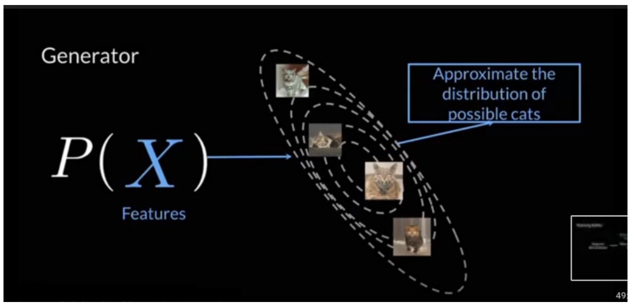


What sampling basically means is we have these random noise vectors and when we input that in to the saved generator it can generate all sorts of different examples.

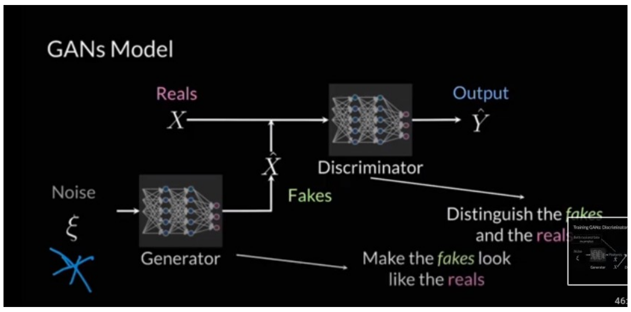
So we can continue generating new noise vectors , putting it through this saved generator and then sampling more dog images.



The most common cat breeds will actually have more chances of being generated because they are more common in the dateset and certain features such as having a point ears will be extra common because most cats have that but then rare breeds will have a less likely chance of being sampled.

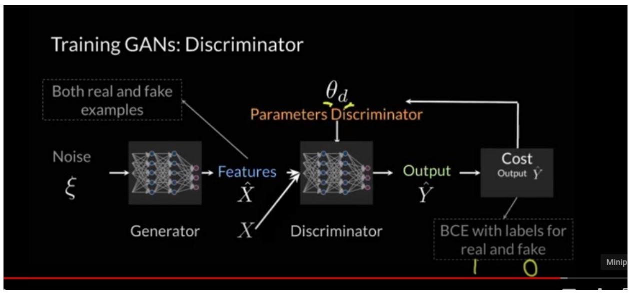


Putting it all together



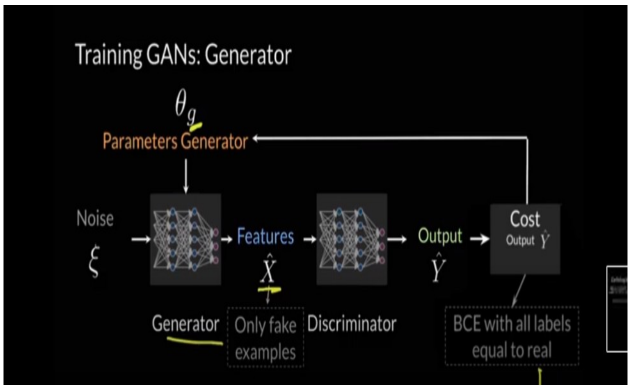
We don’t need to pass a class to the generator but we can if we are generating a lots of different classes.

So to train a basic GAN we alternate the training of the generator and the discriminator.



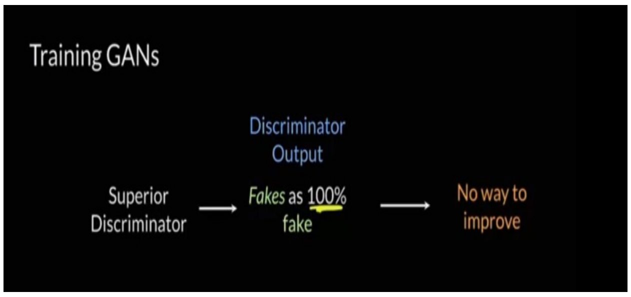
First we get some fake examples X produces by the generator from that input noise and those examples the fake ones X and the real ones X are both passed in to the discriminator without telling the discriminator just yet which ones are real which ones are fake and then the discriminator makes predictions y of which ones are real and which ones are fake or more specifically the probability a score of how fake and how real each of these images are. After that the predictions are compared using that BCE Loss with the desired labels for fake and real and that helps update the parameters of the discriminator.

So this only updates the parameters of the discriminator , only this one neural network , not the generator. For the Generator , it first generates a few fake examples and this will again be passed to the discriminator but in this case the Generator only sees it's own fake examples , it doesn't see the real examples at all and then the discriminator makes predictions y of how real or fake these are and after that the predictions are compared using the BCE Loss with all the labels equal to real Because the generator is trying to get these fake images to be real or label of one as closely as possible. so after computing the cost the gradient is then propagated backwards and the gradient is then propagated backwards and the parameters of the generator or theta(g) will get updated.



Now only the generator , this one Neural Network that's going to updated in this process not the discriminator. so as we alternate the training only one model is trained at a time and the other one held constant.

We need to keep in mind that both models should improve together and should be kept at similar skill levels from the beginning of training. so the reasoning behind these is if we had a discriminator that is superior than the generator , we will get a prediction from it telling us that all the fake examples are 100 % fake , well that is not useful for the generator , generator doesn't know how to improve , there isn't anything to know which direction to go in and learn over time.



And also if we had a superior generator that completely out scales the discriminator , we will get predictions telling us that all the generated images are 100% real , so when training GANS in these alternating fashion , it's important to keep in mind that both models should improve together and should be kept at similar skill levels from the beginning of the training.

Largely because of the discriminator , the discriminator has a much easier task , it just trying to figure out which ones are real and which ones are fake as opposed to modeling the entire space of what a class would look like , so the discriminator's job is much easier than the Generators , one common issue is having a superior discriminator , which learns too quickly and when it learns too quickly it suddenly look at a fake image and "says this is a 100% fake " but this 100% is not useful for the generator at all because it doesn't know which way to grow and learn

Batch Normalization

- What is Batch Normalization
- What do we understand from the name ?
- Why do I need Batch Normalization
- What is the problem that we faced which enables us to create Batch Normalization ?
- How does the Math for Batch Normalization works ?
- How do i know if i need Batch Normalization
- How to implement Batch Normalization in PyTorch

Wassserstein Generative Adversarial Networks with Gradient Penalty