

**Medical Image Analysis**  
**Professor Ganapathy Krishnamurthi**  
**Department of Engineering Design**  
**Indian Institute of Technology, Madras**  
**Lecture 49**

**Generative Models**

(Refer Slide Time: 00:23)



**Introduction**

MRI Medical Imaging & Reconstruction Lab



- Data is in the form of images, videos, text and tables numbers are used for drawing inference - Machine Learning techniques
- Generative models use probabilities to describe/characterize data
- Any data  $D$  can be seen as samples from a probability distribution  $p_{data}$
- Generative models try to approximate this distribution given the training data  $D$
- The learned model can then be used for inference tasks

Hello and welcome back. So this will be the last week of classes where we are going to look at generative models, specifically generative adversarial networks. So we have usually seen data in the form of images, videos, again moving images, texts, tables, numbers, etc. And these numbers are used as input for drawing some inference. And a variety of machine learning techniques are geared towards, were given an input, or provide an output in the form of a classification or some regression value.

So generative models are models that actually describe your data, so generative models try to model the distribution of the actual training data that is given to you. So that way, you can say that generative models use probabilities to describe or characterize your data. So the data set  $D$  that is generally made available to you, for training your deep neural networks can be seen as samples from a probability distribution,  $P_{data}$ , which is the true distribution of data, this distribution is generally hard to get because you actually do not have access to all possible kinds of data corresponding to the problem that you are working on.

So for instance, you are working on images of the brain, MRI images of the brain, you would need access to pretty much all possible images of the brain. So every person on this planet,

the generally not available, so you have access to a small subset data of the data  $D$ . But then you see that data has been drawn from this probability distribution  $P$  data, which you would ideally construct using all the infinite number of samples. Typically, that is what you would look at.

But then you have this subset of that a small subset of that, that is made available to you for training, and you guys actually, make an empirically distribution or  $P$  data, or  $P$  data that, that that is the real distribution. So  $P$  data is real distribution, you would make an empirical distribution from it using your training data set.

So the what is the idea behind generative model? Generative models try to approximate this  $P$  data. Given the training data  $D$  using neural networks using these neural networks in general, you can say that using differentiable functions, but in this case, they are going to be deep neural networks.

What is the idea behind this? Well, once you have this idea is that you can then draw samples from  $P$  data using the neural network. So either the neural network provides you with the parameters, or the distribution itself that some models do, or indirectly give you samples from the distribution, which is what GANs does.

So given and, it actually depends on one latent noise factor that you provide as input, we will see that over the next few slides. But these are different kinds of generative models that you can have. So to summarize, you have the real probability distribution,  $P$  data, this generally not available to you, and you will not have access to all the data itself. So you only have access to training data  $D$ , which is representative of the problem you are trying to solve. And what you are trying to do is to get a neural network to approximate this distribution  $P$  data.

(Refer Slide Time: 03:24)



MRI Medical Imaging & Reconstruction Lab

## Generative models

- Parametric approximations to data distributions are good generative model- for e.g. if we can model given data as a Gaussian distribution, we can describe the data using the mean and standard deviation
- The distribution can be sampled to generate new data points and the probability of a particular sample  $x$  can be easily determined for inference
- The model parameters are learnt by minimising an objective function - a distance metric which measures the distance between distributions

$$P_{\text{model}} \quad P_{\text{data}} \sim \tilde{P}_{\text{data}} \quad \text{KL divergence}$$

So what is the easiest thing to do? Easiest things are parametric approximations to data distributions they will serve as good generative models. So for instance, if we can model our data as a Gaussian distribution, then we can just describe the data using the mean and standard deviation. And subsequently, we can also we also know how to sample from this different distribution.

So given this kind of Gaussian distribution, we would know how to sample it. Now, the problem to be solved in such cases is to estimate the parameters of the distribution, which is again, done, you have no by minimizing a or maximizing an objective function. So what you had? Yes, you will have a  $P$  model.

And then there you have this  $P$  data, you actually do not access the real one, but you would have this  $\tilde{P}$  data, which you would construct using the training data set  $D$  and then you try to minimize the distance, a distance metric between these two typically domestic that you would minimize would be the KL divergence, Kullback–Leibler divergence, which is what is used in a lot of the estimation problems.

So this is one way of doing it, and what we are going to look at is a particular technique, which was which was published in 2014, which turned out to be very popular way now, it is also used very often to provide generative models using deep learning.

(Refer Slide Time: 04:51)



## Learning Generative Models

MRI Medical Imaging & Reconstruction Lab



$$P_{model} \sim P_{\theta}$$

- Distance between the desired distribution and the training Data distribution
- $P_{\theta}$  is the desired distribution parametrised by  $\theta$ , and  $\theta$  is estimated such that  $d(P_{\theta}, P_{data})$ , where  $d()$  is a distance metric between probability distributions
- How to estimate  $P_{data}$ , for let's say a table of numbers or for an image set like ImageNet. Challenging?

$$256 \times 256 \sim 10^8 \text{ parameters}$$

So how do you do this? How do you learn this generative models? What you have to do is find out a distance metric between the desired distribution and the training data distribution. So, the sense you have your model we call this P model, or you can also call this  $P_{\theta}$  because theta are the parameters of the distribution.

So, for instance, if you are using Gaussian is parameterized by  $\mu$  and  $\sigma$ , which Bernoulli you have, you have the Bernoulli parameter. And we want to estimate this  $P_{\theta}$  such that, you know, there is some distance metric  $d$  between  $P_{\theta}$  and  $P_{data}$ , which is optimized as a maximized or minimized in this case, minimax would be more likely.

But then, you know, it is not so easy, as it said, because as the dimensionality of the problem increases, you are in trouble. So for instance, if you are trying to, let us say, approximately know, find a generative model for images, so a  $256 \times 256$  image will have at least  $10^4$  to  $10^5$  parameters, one for every pixel in it, so that is a problem. So the number of parameters will have an explosion in the number of parameters, you are trying to do this in a very straightforward brute force manner. So that is why this problem is generally very hard to do.

(Refer Slide Time: 06:11)



## Learning Generative Models

MRI Medical Imaging & Reconstruction Lab



- A typical medical image has about a million ( $10^6$ ) voxels or more.
- If voxel values were quantised to take values  $0-255$ , that gives rise to an exponentially large number of images- but amount of data is limited
- However, image data has structure that can be extracted, we can call this structure features - priors
- Better to learn these features automatically rather than defining them for every data set- Deep learning

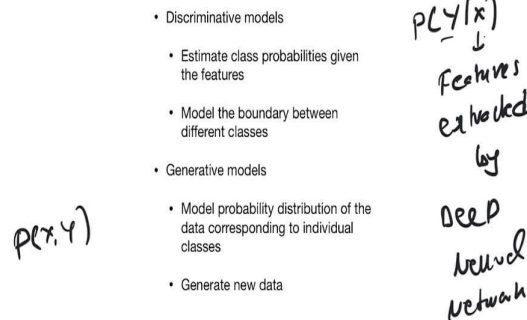
So like I said, a typical medical image will have about a million voxels or more. And even if we quantize them to take big values between 0 to 255, remember that a lot of the medical images can take a much more wider range of values. And given this you have this means that every pixel can take 255 values around millions of pixels. So you can see that gives rise to exponentially large number of images, but the amount of data that you have to approximate this distribution is very limited.

However, what is possible is that this image data has some structure to it, which can be extracted automatically using deep learning. And that is also another fact that we exploit when using generative models based on deep learning.

(Refer Slide Time: 06:56)



## Discriminative vs Generative



So what have we seen so far in terms of, we are going to use neural networks for analyzing medical images or in general images, so what we are seeing is we are looking at classifying pixels, classifying images. Now, these are typically known as discriminative models, because what you are trying to do is, what your output is basically is the probability of belonging to a certain class, given the data.

Now this  $x$  is basically features extracted by the neural network, by the deep neural network. So that is typically what you are doing all the time, when you are training the network to either do semantic segmentation or image classification. So you are just training a discriminative model. So, what it does is estimate class probabilities given the features  $P(Y/X)$  and it also one way of saying that it models the boundary between different classes.

Now, what generative models do? Generative models model probability distribution of the data corresponding to individual classes, so it is like you can think of it as modeling  $P(x,y)$  a joint probability distribution can be model. So from there you can generate new data. So if you know  $y$  can we get  $x$  completion etc.

So, all of these are possible with generative ones. So, now in the recent past, you know, an explosion in the sophistication of these models. So, all those stuff you read about on social media, how images are generated, art is generated, so, you input text. And think of and then from there from the text images are generated, and you input a cartoon and then pictures are generated, and our faces are generated automatically.

So, all of those are generative models and they have a wide range of applications. So, in the case of medical image analysis, what would you have, what do you see as an application, in


medical analysis usually there is a paucity of data especially labeled data. So, in that case, if you can come up with synthetic data that can be used to train deep neural networks, so that they can be deployed everywhere.

So, the advantage of synthesizing data is that you can synthesize a wide variety of data, maybe also tried to model you know, how data varies from different centers or different scanner will that images process and try to generate data that is realistic, and it will be trained, will say for classification, and then can be deployed in any hospital or center.

So that is one of the one possible application here of generative models. So, like, so, likewise, you can you can also try to simulate different types of pathologies, all of that is possible only using generative models of data.


(Refer Slide Time: 09:44)

# Applications

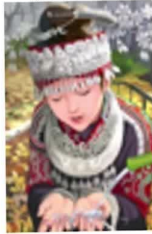


- • Super resolution ←
- Colourisation of grey scale images
- Predicting occlusions in images
- • Cartoon to image conversion, image modifications
- • Generative models of speech, text to speech


original



bicubic  
(21.59dB/0.6423)



SRResNet  
(23.44dB/0.7777)



SRGA  
(20.34dB/C)




Figure: Photo-realistic Image Super-Resolution Using a Generative Adversarial Network, by Lediq et al.

So in the context of you know, vision, and in general, if you think about any application, here are some applications. So for instance, the application that you would consider super resolution this is again, in medical imaging, this is very useful. So how will you use it, so for instance, we are to get very high quality images it is from a scanner, typically, let us say CT or MR scanner, the price of the hardware also goes up.

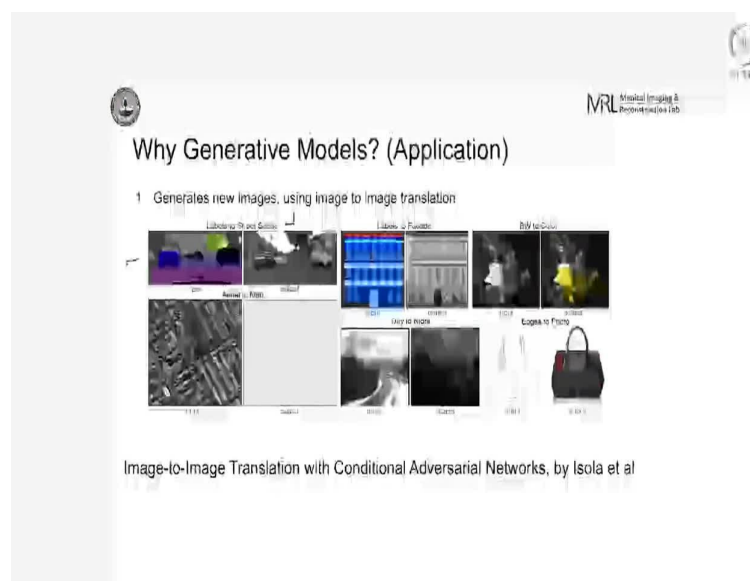
So if you are able to set up on deep neural network, or a generative model that can take low quality data as input and generate high quality, high resolution data, then you can actually, make the scanners economical. So that is one very broad goal that that can be for a deep or deep learning generative models in medical image analysis.

The others are very useful in vision colorization of grayscale images, predicting occlusions in images where and if you have damaged paintings, etc, we will try to fill them in. Now cartoon to image conversion, which you can use for designing or, even creating works of art, and of course, there are always generative models of speech, text to speech, etc.

So somebody speaks, you convert that to text, somebody types it in and you convert that to speech and automatic speech generation, you can call it that. So in all of these, you know, there is always an input, if you notice, for super resolution, you need to have a lower resolution image of the same object, etc. That is fine.

So, it is still generative, because you can think of that lower resolution object as somewhat what they refer to as a latent variable as this hidden variable, which actually describes you I know is a lower dimensional description of your real high dimensional data. So all of that is fine. But many of these are or applications both in computer vision, as well as in medical image analysis.

(Refer Slide Time: 11:40)



So there are again, multitude of applications, you just have to do a lot of literature survey, you will see that, image to image translation. So if you give a semantic segmentation as input, it gives you a realistic scene. Again, once again, this can be used even for medical imaging. Of course, the problem with medical imaging data is that three dimensional inherently 3D, most of the CT and MRI scans and, you know, getting anatomically correct, anatomically correct generation of images is a challenge.



(Refer Slide Time: 12:17)

 **Applications** 

- Generative models of faces - various models over the last 5 years have seen improved results



LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS  
Andrew Brock, Jeff Donahue, Karen Simenon



↑ [StyleGAN, Karras, Laine, Aila, 2018]

→ Image guided surgery

So there are multiple such applications of the recent past, I am sure you all seen in media or social media, huge photorealistic. So if you see this particular, mosaic of images, all of them are fake, pretty much, they are not real people. Similarly, on the left, all these images are generated from deep learning models.

Once again, they have a lot of applications, both, for in media as well as computer vision, in the context of, medical image analysis, as like you already mentioned, it would be great if we can generate, let us say, anatomically correct 3D volumes, so that we can use them for training. Of course, there is a further challenge annotation is also a challenge. There are variants of this to a hopefully, we can also have annotations automatically.

The other application for GANs, for instance, generative models in general, in the field of medical image analysis is in you know, guide, image guided surgery. So again, I am just mentioning these applications. Of course, these are lot of these are in the research stage for a lot of publications in this area.

So image guided surgeries, wherein, you know, most of the time you are intra operative imaging system is just a ultrasound imaging system. So during surgery, if you want to monitor, your surgical tools usually have an ultrasound, you know, or just plain camera sometimes. But your 3D high resolution image of the anatomy comes from MRI CT, which you took before the surgery.

So now, if you want to align, we looked at image alignment or image registration, you want to align the images that you take during surgery, that ultrasound, mostly, with the images that you acquired prior to surgery, which is basically high resolution, then it would be nice if both of them are, the same type.

So now you are going to have to do cross modality image registration is generally a difficult problem. So it will be nice if you can convert the interpret images, basically your ultrasound images into MR, so that the registration problem becomes much simpler. So for such applications, also, you can use generative models.

(Refer Slide Time: 14:24)



MRL Medical Imaging & Reconstruction Lab

## Image Translation



[CycleGAN: Zhu, Park, Isola & Efros, 2017]

<https://github.com/junyanz/CycleGAN>

Of course when I say that I know these are again, in the research stage, the much difficult problems to solve also, once again, here I am displaying this image translation where in your time you have based on a certain type of training, you can convert this image of a horse into that of a zebra.

(Refer Slide Time: 14:43)



2. Generating speech from text



MRL Medical Imaging & Reconstruction Lab



WaveNet: A generative model for raw audio, by Van Den Oord et al.



Generating speech from text there is a WaveNet. You can look this up. Once again, we are not going to cover this in this class, but it is a very popular model.

(Refer Slide Time: 14:54)



Generating Sequences

Medical Imaging & Reconstruction Lab



would find the bus safe and sound  
As for Clark, unless it were a  
can near at the age of fifty-five  
Editorial. Dilemma of  
the the tides in the affairs of men;

Generating Sequences With Recurrent Neural Networks, by Alex Graves

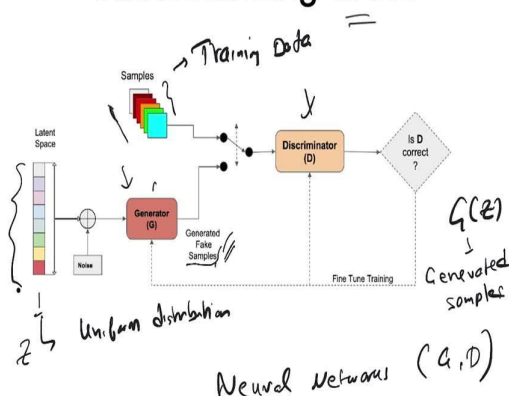
Similarly, generating sequences, a lot of the language models you can consider them a generative because you just have to provide an input starting sentence and complete sentence. For instance, even image translation is generative. You provide an image as input in one language and the output comes in another language. All of these are basically generative model.

(Refer Slide Time: 15:16)



## Understanding GANs

Medical Imaging & Reconstruction Lab



So the model we will be talking about in this class is the generative adversarial networks by Ian Goodfellow, it is a very popular work in a popular work, I mean context of lots of citations. Of course, a lot of other publications have come off following this publication of GANs, a generative adversarial networks. And we are going to look at this briefly.

Because you are going to have a slightly fairly long tutorial session on, a neural network implementation deep neural network implementation of this. So how does how does GANs work? How does it generate images? That is the, that is what we are going to look at in the following slides.

So basically, this particular generative model consists of two parts or two pieces so to say, two functions. Which one is a generator, the other is a discriminator. Both of them can be neural networks. The generator and the discriminator both are neural networks. So what does it do? So first, let us just look at the generators work, and then we will look at discriminator later.

So a generator takes as input a latent space vector. So this latent space vector is often denoted as  $z$ . And, basically, and that  $z$  is you can also can and sometimes it is also referred to as noise, what are and that  $z$  is slight, most of the time, slightly lower dimensional than the data you are trying to generate. But ideally, it should be the same size as the data you are trying to generate.

So that is typically what is done, you will see that in the paper, the work through that we have in this week. So this  $z$  is input, and idea is called latent space, and to give you an understanding of what this might mean. So for instance, if you if you think of face recognition, our face recognition, what would this latent vector be, it would be maybe a distance of one particular vector with distance between the eyes, distance between the ears, now width of your lips, the angle that your jaw makes the vertical or the horizontal, the width of your temple, color of your hair coded in some form, so all of them in some form of numbers, that would help you construct reconstruct a face, all that data that you can think of visually, as will help you reconstruct a face accurately. So that will forms it, that is a good way of understanding what  $z$  is.

If you sample  $z$ , the idea is from  $z$ , you should be able to recreate samples of your distribution. So that is what this function does the generator is a neural network, I call it a function, it is a differentiable function, it takes us input  $z$ , and outputs is  $G(z)$ . This  $G(z)$  is what you call a sample, is a generated sample.

Sample that is generated by the function  $g$ , which is which is in, currently, all of those are, all of  $G$  are just basically neural networks. So  $G(z)$  is the output from  $G$ . So these are generated samples, also referred to as generated fake samples so these are fake samples. And then you

have your training data, your training data distribution, these are samples of the training data, you should have access to training data, by the way, you cannot do this out of thin air.

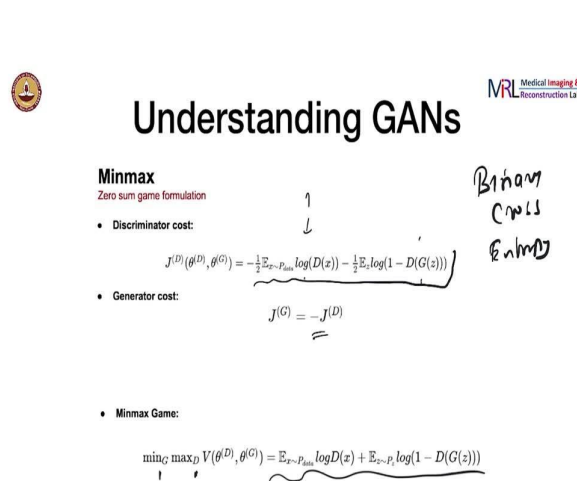
So you should have some access to training data,  $G$  simulated or some, some collected, some amount of it is an acquired. So you take the samples of training data. And you also by, you know, giving a bunch of latent space variables. So this latent space can be sampled from a uniform distribution, multi-dimensional uniform distribution. Think of it as multi-dimensional uniform distribution or you sample a bunch of numbers from there.

So from uniform distribution, and for each one of your latent vector that you have sampled, you have an image generated. And that gives you a bunch of images and you have samples from your training data. So what does this discriminator do? Discriminator is trained this way. So you have 50% of the samples, which are input to the discriminator coming from the training data, 50% of the samples input to the discriminator coming from the fake data.

So let us say 50 images from samples 50 fake data and the function of the discriminator is to classify them as 0 or 1, 0 corresponding to those generated by a generator, which you call fake samples and 1 corresponding to the real data. So it is like a two class problem in this case, you assigned it to one of either classes or real data or the fake data. Fake data corresponds to 1 coming out from  $G$ . Real data is from the training set, that is the discriminator.

So the way this is set up is you have now have these two functions, discriminator and generator, which are basically neural networks. And you have to train these neural networks to estimate the parameters of the neural network. So we will see how such training is done in the next few slides.

(Refer Slide Time: 20:32)



**Understanding GANs**

**Minimax**  
Zero sum game formulation

- Discriminator cost:
 
$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log(D(x)) - \frac{1}{2} \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))$$
 Binary Cross Entropy
- Generator cost:
 
$$J^{(G)} = -J^{(D)}$$
- Minimax Game:
 
$$\min_G \max_D V(\theta^{(D)}, \theta^{(G)}) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))$$

So here is the loss function that is trained for the for training GANs, so the discriminator and generators have slightly different loss well have slightly different loss functions, but they vary in sign. So the idea is the following. For the discriminator loss function, it is very similar, very similar to the binary cross entropy so very similar, I will not say same, because what happens is, you know, all the positive samples, the 1 labels are drawn from the training data, all the 0 labels are drawn from the  $G(z)$  or basically the 1 that are generated by the samples.

So by this is how you train the parameters of the data of D, or data of D or the parameters of the discriminator, the generator cost is exactly the opposite of it. Because you would want to fool the discriminator, the generator is actually trying to pass off the samples it generates as real samples or samples that come from the training distribution to the training data. So it is generally so it is cost you can say, is basically the opposite of what the discriminator is trying to do.

So it will try to make it will try to make the discriminator do more mistakes in the sense it should, it will classify fake data as real. So that is why you have the minus sign on it. Of course, when you are doing this, this this term disappears, the expectation or the training data disappears, because you do not actually consider it, because you are only looking at the images that come from  $G(z)$ . So that is the generator costs.

So when you have something set up like this, it is called the zero sum game or minmax game. So basically, you are maximizing over the parameters of one network and minimizing the

over the parameters of the other network and this particular loss function. So we will look we will look at it in slightly more detail over the next few slides.

(Refer Slide Time: 22:34)



$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2} \mathbb{E}_{x \sim P_{data}} \log(D(x)) - \frac{1}{2} \mathbb{E}_z \log(1 - D(G(z)))$$

$y = 0 \rightarrow$  output of Gen - fake  
 $y = 1 \rightarrow$  Training data - real

$D(x) \sim 1 \rightarrow y \rightarrow 1$	$y \rightarrow 0$
$D(x) \sim 0 \rightarrow y \rightarrow 0$	
$D(x) \sim 0 \rightarrow y \rightarrow 1$	
$D(x) \sim 1 \rightarrow y \rightarrow 0$	

$\min J^{(D)}(\theta^{(D)}, \theta^{(G)})$

So let us look at the loss function for the discriminator first, so, the labels the true labels are 0 and 1. As a true labels, so which means that  $y$  is 0, meaning output of generator which is fake. And 1 meaning from a data from training data, this is training data sample, which means it is real, this is fake, this is real. So let us do let us look at the case when the discriminator works properly.

So, when discriminator works properly means  $D(x)$  should be close to 1, when  $Y$  corresponds to 1 that is training data, and  $D(x)$  should be close to 0, when  $Y$  corresponds to 0, because these are the class labels. And if you do that, then you will see that  $J$  takes the value close to 0.

Now, when it misclassifies, so when it misclassifies when everybody when we miss classification, that when we see a different color, so  $D(x)$  outputs 0, when  $Y$  is 1 that is it miss classify if real, as fake, and it misclassifies fake as real when  $y$  is 0, so you can plug these values back in here.

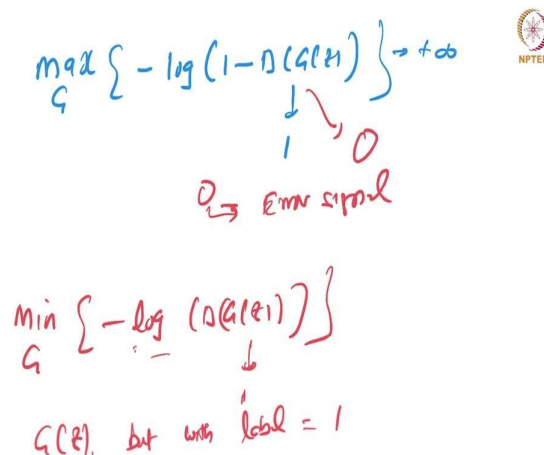
So when  $D(x)$  goes to 1, and a sense,  $D$  of  $G$  of  $z$  this, forget this one will get forget this particular, when you are getting data from  $G$  of  $z$ , let us forget this. And if  $D$  of  $G$  of  $z$  turns out to be 1, this will go to minus infinity, so minus of minus infinity will be plus infinity.



Similarly, when you are getting real data, this term will not be there. And that is data from the samples training samples.

So and if you are misclassifying, then this will be 0 log of 0 will be minus infinity. So this minus will make it plus infinity. So the way to make the discriminator learn would be to minimize this loss function. So you would minimize  $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ . Once again, theta G refers to the parameters of the generative model, which is another neural network theta of D refers to parameters of the discriminative model which is another neural network.

(Refer Slide Time: 25:04)



The image shows handwritten mathematical expressions for the generator and discriminator loss functions. The top expression is  $\max_G \{ -\log(1 - D(G(z))) \} \rightarrow +\infty$ , with a red arrow pointing from the  $1 - D(G(z))$  term to a red '0' and the text 'Error signal'. The bottom expression is  $\min_G \{ -\log(D(G(z))) \}$ , with a red arrow pointing from the  $D(G(z))$  term to the text 'G(z), but with label = 1'. An NPTEL logo is visible in the top right corner.

So what happened to the generator? So the generator is slightly more tricky. So what we want to do is if you look at it, we did it to be the negative of the discriminators loss function. Now in the discriminators, discriminators loss function there was this term which had the expectation over the training data we do not need that because for generator it is only going to deal with samples from the generator.

So, the way it will you would do that is you would do the following optimization maximize over the parameters of G following minus log of. So, what would you do? So, if what it does is the following let us say when this generator does an excellent job it generates a beautiful image or just like the images in the training distribution then  $D(G(z))$  will become 1.

So,  $-\log(1-1)$  is 0 it will go to  $-\infty$ , so,  $-(-\infty)$  that will be tending towards  $+\infty$ . So, now what happens if the discriminator missed correctly classifies, correct. So, which means that D of let me correctly classifies then  $D(G(z))$  becomes 0, but the log 1 will go to 0.

So, this is basically your error signal that can be brought back propagated, so you actually want I know signal from here when you know to improve the generator. Because if the discriminator is able to correctly classify the output coming from the generator as fake, then that is not good for a generator.

You would try to use that signal or to modify the parameters of the generator but and it turns out that in this heuristic, it kind of becomes hard. So, what actually implementation is you would flip the labels. And then minimize the following loss. So, the minimize session of the parameters of G, you would do the following minus log of D(G(z)).

So, what happens here is that if you notice, when D misclassifies in a sense, it is correctly classifies G(z) as belonging to fake. So, it means 0. So, then this loss function gets a maximum value, so, plus to infinity and if D(G(z)) if D misclassifies it as true, then if this output is 1 log of 1 is 0, so, you get 0, that is the minimum value.

So, you want to minimize this loss function. Basically, you want to you want to force the discriminator to make mistakes. In this case, the mistake is classifying the fake image as real. Now, this is the same as you know, inputting the image, G(z) but with label equal to 1. Remember, we said the real image was labeled 1 that has images from the distribution of label 1. And images from the, generated by the generator was labeled 0. So this is what we do is you use the this loss function, but we give the label as 1 and so this is that is how you would train the generator.

(Refer Slide Time: 29:07)




---

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator,  $k$ , is a hyperparameter. We used  $k = 1$ , the least expensive option, in our experiments.

---

```

for number of training iterations do
  for  $k$  steps do
    • Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
    • Sample minibatch of  $m$  examples  $\{x^{(1)}, \dots, x^{(m)}\}$  from data generating distribution  $p_{data}(x)$ .
    • Update the discriminator by ascending its stochastic gradient:
      
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(x^{(i)}) + \log (1 - D(G(z^{(i)})))]$$

    end for
    • Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
    • Update the generator by descending its stochastic gradient:
      
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)})))$$

  end for

```

---

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

---

So here is a brief look at the algorithm from the paper. So, they actually train the generator and the discriminator alternatively, so, for instance, you would sample mini batch of  $m$  noise samples from a  $P_g(z)$ . So this  $z$  is just a vector of uniform distribution typically, and then you take the mini batch of  $m$  examples from the data and you update the discriminator.

Similarly, sample mini batch of  $m(y)$  sample some noise prior and you update the generator. So, you can do this by the finish the generator and discriminator so on so forth and you can alternate. And that is how you would train these networks. So following this we will look at so called convolution based generative adversarial networks adversarial networks which will be part of the tutorial, thank you.