

Machine Learning for Engineering and Science Applications
Professor Dr. Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology, Madras
Generative Adversarial Networks (GAN)

(Refer Slide Time: 0:14)

Generative Adversarial Networks (GAN)

- Class of generative models
- No explicit model but allows one to sample the model distribution
- Sampling is done using a deep neural network
- The neural network takes as input random noise and transforms it into the model distribution

$P_{data}(x) \rightarrow \{x_i\} \quad i = 1 \dots N$
 $P_{model}(x) \sim P_{data}(x)$
↳ Deep Neural Network

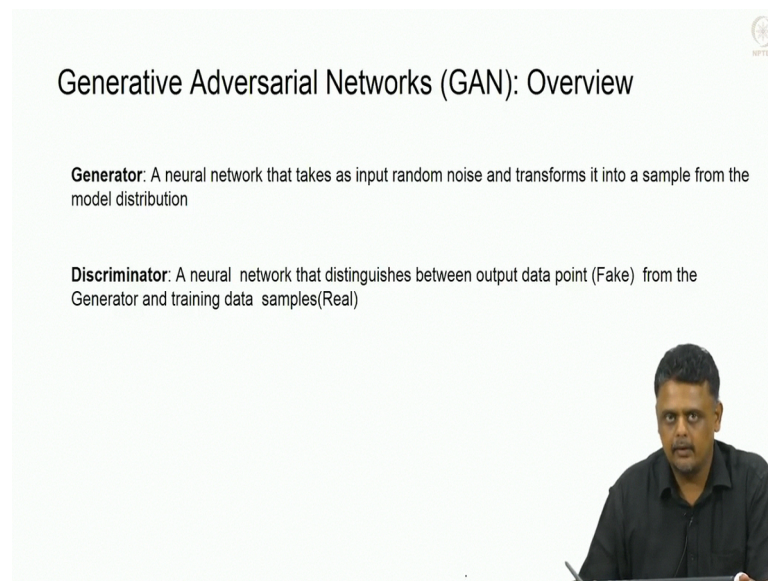
Goodfellow et al, Generative Adversarial Networks, Advances in Neural Information Processing Systems, 2014

Hello and welcome back, in this video look at generating adversarial networks, these are class generative models which do not explicitly model the data distribution, but rather provides a sample for it and sampling is performed using a deep neural network, the neural network which actually provides a samples, takes as input a random noise vector and then maps into a sample of the model distribution.

So let us say we have given, you are given training data right, V data of X , which is in the provided based on the samples provided, so X_1 where I is 1 to N , what we want to do is to model some provide a, to want to do is determine this model, P model of X , so that it is an a good approximation of P data, the true distribution okay, so this period of X , once again, we do not actually haves access to all the data that is possible, so we only samples from P data of X and we want to determine some P model, which basically the probability distribution of X .

A model for that, so that we can sample from it, now in this case we do not explicitly model, so in the sense it is not a parametric model, but rather this is accomplished using a deep neural network, neural network actually generates a sample from the model distributions, so we will see how this is done.

(Refer Slide Time: 1:59)



Generative Adversarial Networks (GAN): Overview

Generator: A neural network that takes as input random noise and transforms it into a sample from the model distribution

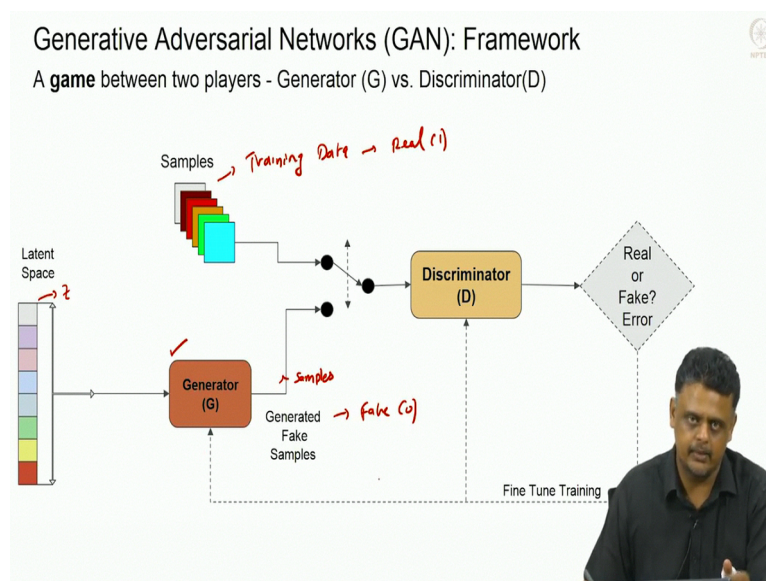
Discriminator: A neural network that distinguishes between output data point (Fake) from the Generator and training data samples (Real)

The slide also features a small video inset in the bottom right corner showing a man in a dark shirt speaking.

The generative adversarial networks, framework consists of two neural networks, one the generator and the discriminator, the function of the generator is to take as input, a random noise vector and transform it into a sample from the model distribution and the discriminators job is to, is actually act as a classifier wherein it tries to determine if its input data X came from the generator, we call them as fake samples or was it from the actual training distribution because of the real samples okay.

It is call adversarial because the generator is constantly trying to fool the discriminator into believing that, into making the decision that input generated by it is from the training distribution, while the discriminator is constantly lying to learn the (θ) (2:59), so that it always determine whether, it is call an adversarial network because it has these two networks generators and discriminators trying to work against each other like as I mentioned earlier, the generator constantly trying to generate samples that will fool the discriminator into classifying it as coming from the training data distribution, so in that process the weights of the generator learns a transformation which enables one to convert the random noise input vector into a sample from the model distribution.

(Refer Slide Time: 3:37)



So just to have illustration of what we discussed, so the generator G takes as input, the generator G takes as input a random noise vector also referred or denoted by Z , also referred to as latent space, these random noise vector is then given the, which access input to the generator G , which then outputs some samples, generated samples of the, which are hopefully, similar to the training data distribution.

The discriminatory D takes as input you are trading data, so again, this are samples or nothing but your training data, which are made available to you, so, for instance, if you are interested in generating faces, then you would have a database, faces of different people and that would be the input of discriminators, so it would be a general-purpose algorithm, it has a specific to some certain task, so the training data. It takes as input trading data, which are labelled as real and the generated fake samples again they are labelled as fakes or we can say this is zero label and this is the one label.

The discriminator alternatively tries to, takes as the input the samples, the training data samples as well as the generated samples and outputs an error function, so basically it is the output of the discriminator, which are basically output say probability of particular sample being real or fake ranging from 0 to 1, now this output is what provides the signal, or the error signal to train the weights of the generator as well as the discriminator.

(Refer Slide Time: 5:18)

MiniMax
Zero sum game formulation

- Minimax Game:

$$\min_G \max_D V(\theta^{(D)}, \theta^{(G)}) = \left\{ \mathbb{E}_{x \sim P_{data}} \log D(x) + \mathbb{E}_{z \sim P(z)} \log(1 - D(G(z))) \right\}$$

$J_G = -J_D$

$J(x) \rightarrow 0-1$
 $J(G(z)) \rightarrow 0-1$

So how is that done? This problem is formulated as a zero-sum game, so to speak, because the generators, if you denoted by J of, J subscribe G as the cost function of the generator, then it is basically the negative of the cost function of the discriminator which is denoted as J subscript D , so the cost function the generator is what is given here basically and this also referred to as value function, it is actually a function of two sets of parameters, one corresponding to the discriminator and other corresponding to the generator.

So this is optimized alternatively, there is an inner and outer loop, so the inner loop is maximizing the this value function with respect to the discriminator network parameters, the outer loop is minimizing this again, the same objective function with respect to the parameters of the generator network, so let us take a closer look at the cost function itself, this is the first is $\log D$ of X , which is nothing but the, let us take the closer look at this cost function.

So if you look at this cost function, which is expectation of $\log D$ of X plus the expectation over Z \log of 1 minus D of, so what this means is that will calculate $\log D$ of X with respect to the trading data samples and you will calculate this term with respect to the samples generated from Z . Okay, so this is very similar to the binary cross center P assuming that there is an equal number of generated images and equal number of trading data samples okay.

So this we will see how this cost function make sense, minimizing this or maximizing this cost function make sense in the context of the generative adversarial network, so here, DG of Z is basically the output of the discriminator when the generated images are or given as input,

so D of X goes from 0 to 1 basically you can think of it as a probability of this particular input sample belonging, either being real or fake and similarly and using the generated samples it would be DG of Z, which again go-between 0 to 1 okay.

(Refer Slide Time: 7:42)

Training Discriminator Network

Before Training

$$\max_D [\mathbb{E}_{x \sim P_{data}(x)} \log(D(x)) + \mathbb{E}_{z \sim P_z(z)} \log(1 - D(G(z)))]$$

$D(x)$ should be 1 $D(G(z))$ should be 0

Real Data

Generated Data

Discriminator Network $D(x)$

1

0

$D(x) \approx 0 \Rightarrow \log D(x) \ll 0$

$D(G(z)) \approx 1 \Rightarrow \log(1 - D(G(z))) \ll 0$

So if you look at this cost function, so when we start training, so if you look at it, the discriminator, ideally the output of discriminator should be 1, whenever X comes from the training data distribution and the output of the discriminator be 0 whenever the input comes from the generator, so initially when the discriminator is not sufficiently well-trained, the weights of the discriminator or not are still random, then let us look at a particular case, this case wherein you have real data here and some the generated data which are given as input, what is shown here in this black hyphenated lines is basically the decision boundary given by the discriminator.

So here there is one misclassification real data, here I write here and there is one misclassification of generated data. Okay, so everything to the left this line, left of this line is basically class 0 everything to the, sorry the everything to the left of this line is class 1 and everything to the right is class 0, now when the discriminator misclassifies, so which is the D of X is 0, when X comes from the training data, ideally the output should be 1 but instead it is a 0 and you can see that log D of X becomes a very large negative number right.

Similarly, when you take the database comes from the generator, ideally the output should be 0 but then if this misclassification then once again D of G of Z is close to 1 which means that

log of 1 minus that becomes a very large negative number correct, so this is the case when the discriminator is not performing optimally.

(Refer Slide Time: 9:30)

Training Discriminator Network

After Training

$$\max_D [\mathbb{E}_{x \sim P_{data}(x)} \log(D(x)) + \mathbb{E}_{z \sim P_z(z)} \log(1 - D(G(z)))]$$

$D(x)$ should be 1 $D(G(z))$ should be 0

Real Data

Generated Data

Discriminator Network $D(x)$

1

0

$D(x) \approx 1 \Rightarrow \log D(x) \approx 0$


$D(G(z)) \approx 0 \Rightarrow \log(1 - D(G(z))) \approx 0$

On the other hand, let say it is trained very well and you see that the samples are have correctly classified, the same input real data, as well as fake data here, you see that this is the decision boundary right here and everything above is class 1, everything below is class 0 which in the case decision boundary is correct, then which is the D of X close to 1, then log D of X is closed 0, similarly D of G of set is close to 0, then log 1 minus that is again close to 0.

So, that now you are cost function varies from a very large negative number minus infinity to a close to positive number, which is like in this case is 0. Okay, so maximizing this cost function makes sense, so that maximizing this will lead to the discriminator performing optimally.

(Refer Slide Time: 10:15)

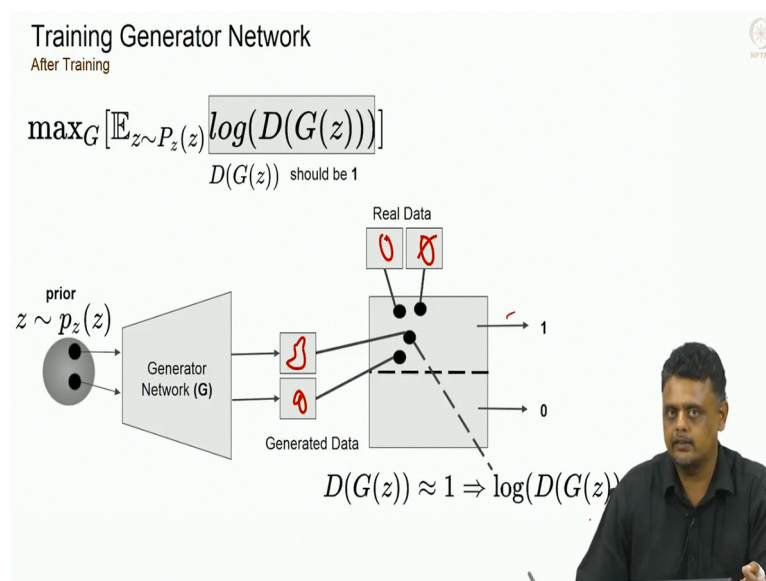
Training Generator Network
Non-saturating Game - Heuristic

$$\min_G [\mathbb{E}_{x \sim P_{data}} (\log D(x)) + \mathbb{E}_{z \sim P(z)} (\log(1 - D(G(z))))]$$
$$\Rightarrow \min_G [\mathbb{E}_{z \sim P(z)} (\log(1 - D(G(z))))] \leftarrow$$


Similarly when you try to look at minimizing the same value function or cost function with respect to the parameters of the generative network, now if you take the first term, it does not have any parameters of the generator networks, so we will not consider that, so will only look at this particular term here, so minimizing this term, what does it mean? Minimizing the likelihood of the discriminator, classifying the fake samples as fake, basically that is what we are trying to do here with this cost function.

So this is minimizing the cost of correctly classifying G as 0. Okay, however it turns out that this cost function saturates very quickly. Okay, because initially when the generated images of very poor quality, then the discriminator has no problems figuring them out as belonging to class 0 okay, so then what happens is that the output of the, the output saturates, so if you take the derivative which is what you raise to the signal that we back propagate to the network, derivative become 0, so there is not much to back prop okay.

(Refer Slide Time: 11:25)

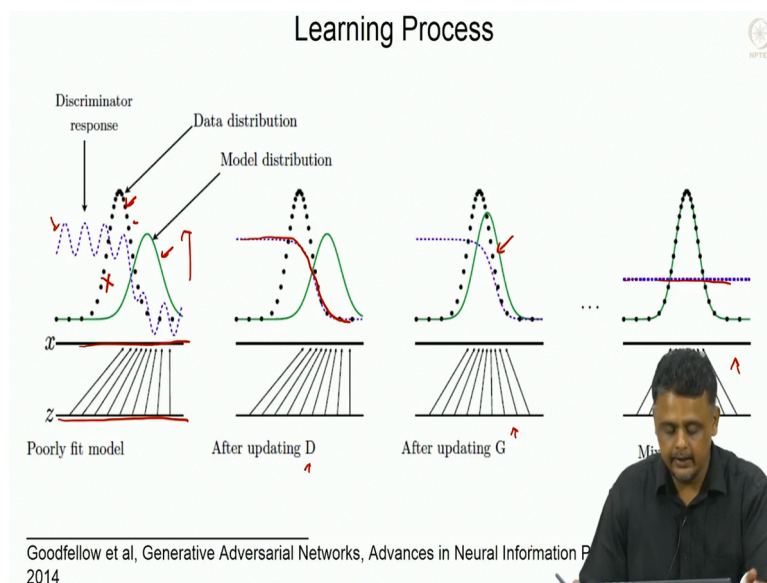


So instead it is placed by maximum of $\log DG$ of Z , so what this does is to maximize the error of the discriminated network, maximize the error in the sense that, it incorrectly classifies D of G of Z as 1, instead of 0, so ideally what we are trying to do is to force D of G of Z to be close to 1 rather than it being close to 0. Okay, that is what this cost function does.

So maximizing the error of your discriminating network is what this cost function does and this provides, this is heuristic and it actually makes it better for training the neural network, so we can just check that is, so once again your real data and fake data being fed into the discriminator, so when the network, discriminator network correctly classifies the output of generator as being close to 0, then it becomes a very large negative number, then \log of D of G of Z is a very large negative number.

On the other hand, when, let say the generator has progress to a point where he is generating, very realistic examples, in that case, let say the discriminator comes incorrectly in this case classifies the output from the generator as belonging to class I, then you know that \log of the of G of Z becomes close to 0. Okay, so then again, once again maximizing this cost function with respect to the parameters of the generator network leads to the generator managing to output samples which are very close to the training get a distribution okay.

(Refer Slide Time: 13:01)



So let us look at this learning process in the terms of one data distribution, this is again from the paper given as recited at the bottom, so let us we have this data distribution in black shown here and let say this is the model distribution from which, which learn by the generated network and this is the, the blue line is the output of the, this is the discriminator response. Okay, we can take this as decision boundary, so initially when the training is not great will see that there is a misclassification by the discriminator okay.

So the misclassification we can judge by seeing that all points to one side of the discriminator decision boundary is classified as real, have poorly points on the other side, let say this side is classified as fake okay, so then after updating the discriminator so you do a several epochs discriminator network and it gave rise to a decision boundary which is much better right now, so which is able to correctly classified samples to some extent from the real versus the generators output okay.

So then what we do is in this case just to have explain further, so said in this X here, this is looking at one-day problem, so this X, this is Z, this is the random vector, this is the space from which we sample the random noise vector, so the generated network maps this to points in X, so X is the data, it wants to data axis and Y axis here, here is the probability okay, here is the probability density function that is what we are looking at.

So Z is mapped to X by generator and it gave raise to this green line. Okay, which is what we are trying to change, so after updating the blue line, the blue dotted line is the discriminator

response, it is getting better and discriminating between the data distribution and the model distribution, however, another epoch of the updating the generated network leads to the green distribution moving closer to the dotted black distribution, which is basically the model distribution is correctly approximating the two data distribution and once training has been and once when both the discriminator and the generator have being optimally train.

Then we get to a point wherein the green and the black dots are coincidental and the discriminator is unable to clearly say which is what, so since that output of the discriminator is always 0.5, ways it is not clear whether the data belongs to the trading distribution, trading data distribution or it came from the generator okay, so this is what the process is, so you alternatively train the discriminator and the generator to point where as an equilibrium wherein the discriminator is not able to distinguish between the samples coming from the training data distribution, or whether it is coming from the generator distribution, the distribution approximate by the generator.

(Refer Slide Time: 16:06)

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 $p_{\text{data}}(x)$.
- 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: **Ascent** $\log(D(G(z)))$

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)})))$$

end for

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



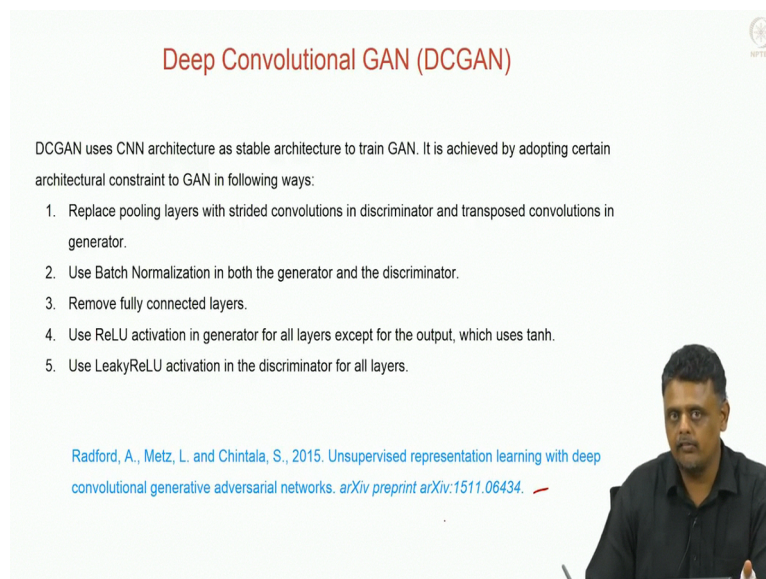
So just to work you do, this is again from the paper and just to work you through the steps involved in the algorithm, so you sample a mini batch of noise sample, so remember the input to the generator or this random noise vectors is of a uniform noise or Gaussian noise, you sample M of them, M is your mini batch size and you also sample a mini batch of M trading data okay.

So again once we sample of course we run it to a generator two, give outputs in the form of the data, so then you update the discriminator by ascending on its stochastic gradient, so we

saw that we maximize the probability of the discriminator when we are trying to train it this cost function and once that is done this again for K types, so this is that loop we are in for K steps and once again we sample a mini batch of M from the noise prior, this is called the noise prior, so this distribution from which you sample Z is called the noise prior and then you update the weights of the generator again doing.

In this case the original cost function is row of 1 minus that, remember that we replace that by now of D of G of Z, so we have to do gradient ascent on it actually, not gradient descent, so this is gradient ascent, to maximize this cost function, so this alternates, so typically you will do 1 step of are several 1 or 2 steps of the discriminator and then go back to the generator and train its weights okay, so in this particular construct, this gradient construct both the discriminator and the generator are neural networks, so usually stochastic the mini batch gradient descent is used for updating the weights of the generator as whereas the discriminator.

(Refer Slide Time: 18:01)



Deep Convolutional GAN (DCGAN)

DCGAN uses CNN architecture as stable architecture to train GAN. It is achieved by adopting certain architectural constraint to GAN in following ways:

1. Replace pooling layers with strided convolutions in discriminator and transposed convolutions in generator.
2. Use Batch Normalization in both the generator and the discriminator.
3. Remove fully connected layers.
4. Use ReLU activation in generator for all layers except for the output, which uses tanh.
5. Use LeakyReLU activation in the discriminator for all layers.

Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

The slide also features a small inset image of a man in a dark shirt, likely the presenter, in the bottom right corner.

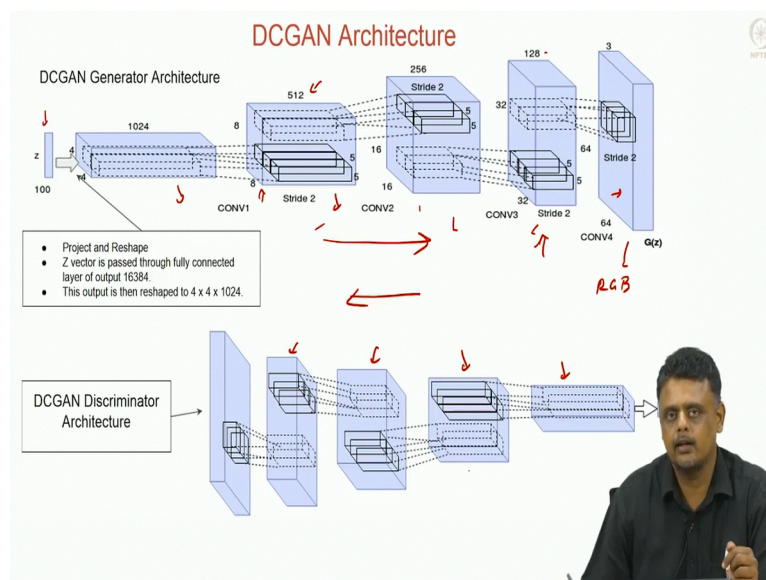
So, will look at one very popular implementation of this GAN its called DCGAN or deep convolutional generative adversarial network, so this is one of the first work inside over, incited here, it is one of the first publications to use a deep convolutional network to generate actual images okay, so the original paper which talks about GANs, used MNIST and it did not use such the deep convolutional networks okay.

So there are some heuristics that they, the authors figure out some of them are listed here, so they replaced polling layers in deep convolutional network with strided convolutions okay, in

the discriminator you have strided convolutions instead of Max pooling and in the generator you have transposed convolutions, remember that we start with a random noise vector and we have to actually generate images, so you need to have transposed convolutions.

Use batch normalization in both the generator and the discriminator, they removed most of the fully connected layers and the use ReLU activation in generator for all layers except the output, which uses a tan hyperbolic and use leakyReLU for the discriminator for all layers, this is particular a heuristic seem to work very well for them, I urge you to read that paper where they have, able to generate images that are not part of the training distribution, but still look very realistic okay.

(Refer Slide Time: 19:33)



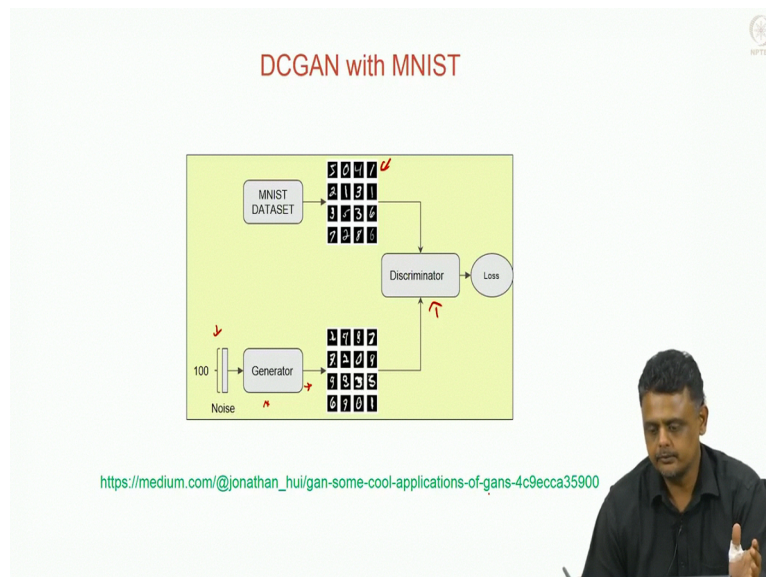
So we will just look at the architecture quickly, so we start with Z , you sample Z from a distribution about 100 points or 100 dimensions and you have to re-project it to a volume of size 1024 feature maps of size 4x4 and then you use strided convolution, in this case they call it fractional strided convolution or transposed convolutions to increase the size of the feature maps to 8x8 at the same time reducing the number of feature maps, this is a typical thing that is seen throughout the networks, so in the next layer you have 16x16 feature maps with 256 feature maps in total.

Then 128x128 feature maps with of size 32x32 and in the end the output is a RGB image right, that is what we interpret the output as the basically 3 channels of size 64x64 okay, so this is the generator network. Okay, so the discriminator is basically the its mirror image basically, so you start off with 64x64x3 and then you go back to this size of 128x32x32,

so on and so forth, so basically what we see in this sequence you go back the same way the okay, all right, so that is what you have here, so all of this comes here and the second comes there, so on and so forth and output is a probability of the image being real or fake okay, depending on what your input is right.

So there are some interesting things in this paper, so basically if you remember that we have to samples Z from a distribution and for generating new images we keep sampling Z , so what they require was they are able to see a continuous transformation of images as you keep changing Z on 1 axis, so the generator was able to meaningfully interpreted between Z , so the paper has some very excellent examples, so you can go and look at them.

(Refer Slide Time: 21:45)



So the idea is again, just to summarize you have dataset of training samples, we will state at using the M is dataset and we have discriminator which is a deep neural network in this case, we have sampling, noise DCGANs paper is 100 dimensions and in the generator takes that are important and outputs MNIST digits in this case, we are just illustrating it with MNIST digits, which is again given as input or discriminator which has a loss function based on predicting whether the, the inputs comes from the training data or whether they are by the output of the generator.

So what is important is that of course you will not be surprised if the generator provides a sound as output samples similar to the one so same as the ones in the training data, however, what is observed most of the GANs-based implementation of the generative models is that very consistently the output images which are similar, but not the same as the ones the

training distribution, these are completely the new images which still makes sense as images and they are able to also interpret meaningfully between the Z values, that is also a very important point to note.

So by constantly, by continuously changing Z you can obtain a sequence of transformational of images which are again meaningful okay, so this property can now has, they have a lot of work has been done in this area right now, I initially, even the paper came out 2014, there was problems in generating larger images, so typically the outputs were restricted to size 64×64 , so on, however, over time, right now there is a something called big GANs which are, which is able to give you very large size images at very higher resolution.

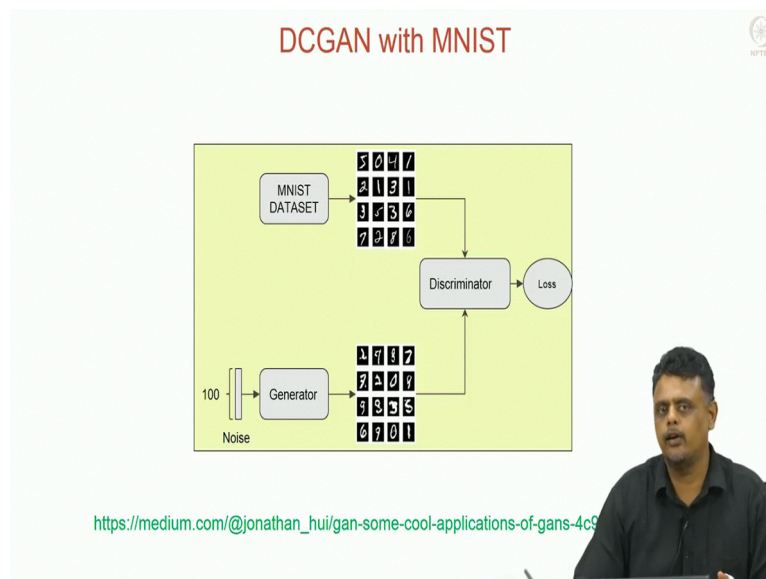
However, of course, the memory required MNIST and the computational requirements of course go up. Okay, so this concludes our session on GANs, we will also look at some applications in medically imaging as to what this generative adversarial networks are used for, in the medical imaging domain, however just briefly GANs have wide variety of applications, for instance especially very at least in the context of medically images, there are situations wherein there is not too much data.

So, in which case you can do the same as supervised learning using GANs, so you can use GANs to generate images like that in your training data set. Okay and all the while training the discriminator for a particular task right, so in this case, the discriminator learns to distinguish between the real and fake images and in the processes, you hope that the discriminator has learned the an underlying representational of your data, which then you can find you, maybe a little bit more data for us, specific segmentation, classification task.

So the context of semi-supervised learning especially for medically images analyses, GANs have wide application, so these GANs can also be used as conditional GANs, in the sense that you can have an additional input, see for both the discriminator and the generator, so that the outputs of both are conditional on the that access input.

So one such application is image translation right, especially, so let say you have two sets of applications, so there are some images which are widely available, let say images of a certain anatomy are widely available in a particular imaging modality, let say MR images are widely available and CT images are not, suppose you have trained a very deep network for MR images right, now you have CT images of the lever but you do not have enough data to train a classifier.

(Refer Slide Time: 26:03)



So what you do is to have a GANs, like network to translate, so GANs have a lot of applications, some of these applications are summarized in this website, it is a very interesting blog, I urge you to go, look at it, however in the context of medical image analysis, there are, this I has very crucial role to play, especially in the context of semi-supervised learning, suppose there is a positive of training data, in fact, label training data, then you can use GANs to train a discriminator which learns underlying structure of the data and then maybe find unit with whatever little data is available right.

Because for training the GANs, you do not probably do not need or label data, just to have access to a lot of images, let say of a particular variety, of a particular anatomy are particular disease and then you can train GANs with it, but since you lack the labels he does not, it is hard to train a deep classifier from scratch, however, once you train a generative adversarial network to generate images for a certain anatomy, then you can take the discriminator and find unit and hopefully it will be a good classifier, so that just one very nice application for GANs.

And there are other fields, like in the case of image translation that there are lot of interest, especially since in the field of medical imaging there are lots of imaging techniques, so MRI, CT, etc and in some anatomy and some disease cases and are more images available with a particular scan, let say there are more MR images of the brain available, or let say more MR images of the, more CT images of lever are available and there is more trading data available, label training data available for CT of the lever other than MR of the lever, then it is could be convey into setup a GANs that work, which can translate CT to MR images, so that you can

label using any classifier you are created for the MR and then translate it back to the CT images.

Of course these are research problems and with the advent of this GANs network, this things are made possible, the recent pass there are being 7 developments something called big GANs as come up which is able to generate larger images because most of the earlier networks following 2014, they have been only able to generate a very small size images, in the sense 64 x 64 or up to 128 x 128 and the resolution was not great either. Okay, so with more progress made in this field, a lot of the interesting problems can be tackled, especially in the field of medical image analysis.