Deep Learning for Visual Computing

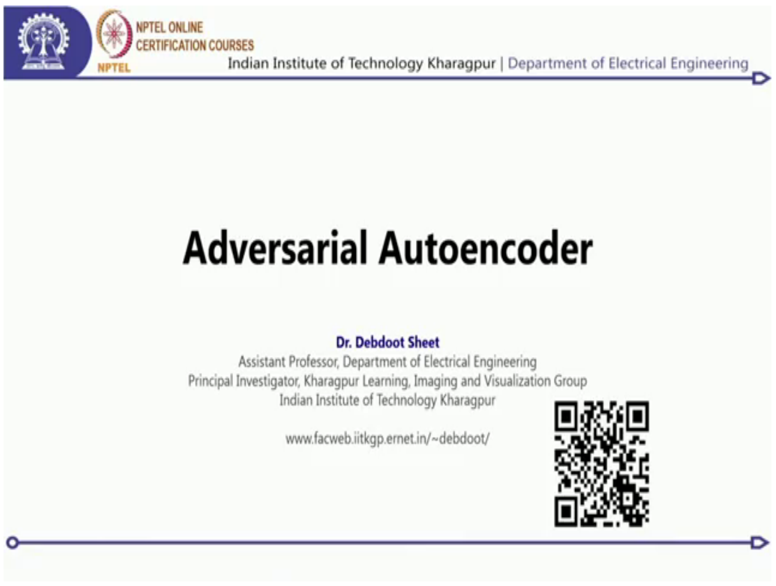
Prof. Debdoot Sheet
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 53
Adversarial Autoencoders

So, welcome to today's lecture. And in the last 2 lectures we have been discussing about understanding the concept of latent variable, and from there how what is actually a latent variable which we called on and we took down a simple example of an autoencoder, and the bottlenecking layer over there, and the output of activations coming out of that as a latent variable. And subsequently we went on to understanding what is the principle of generative modeling or the principle of sample generation over there. And a bit of the math and how can we go on with generating.

So, one example which I took was the mechanism of training an adversarial autoencoder in which the thing was a twined in process, and the training process over there was pretty simple as we had explained out.

(Refer Slide Time: 00:53)



NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Adversarial Autoencoder

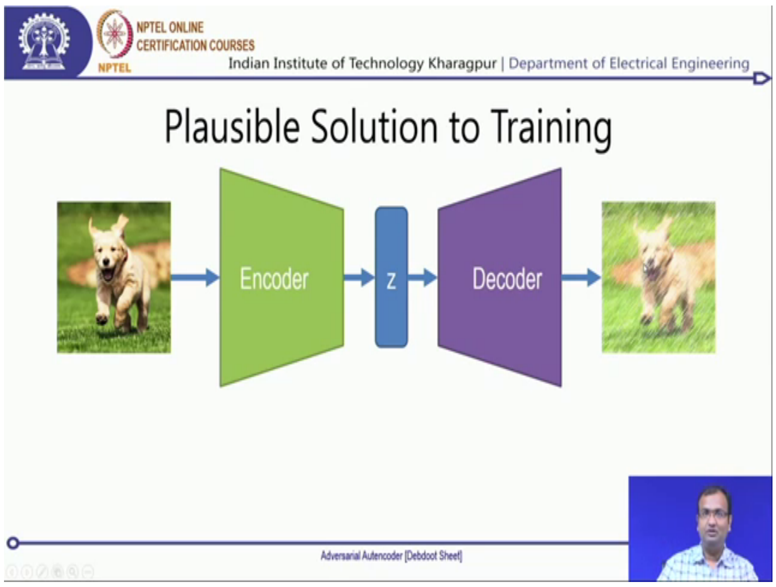
Dr. Debdoot Sheet
Assistant Professor, Department of Electrical Engineering
Principal Investigator, Kharagpur Learning, Imaging and Visualization Group
Indian Institute of Technology Kharagpur

www.facweb.iitkgp.ernet.in/~debdoot/

Now you have an encoder decoder network in which we are going to put an input image, you get an output coming out of it. And on the other part of the Epoch you are going to put a random number over there, and make a discriminator to do it. So, that is where I would be starting with this lecture over here to get into more depth of an adversarial autoencoder and some of the variants. And look into a very practical example of what are the kind of samples which keep on now getting generated using this kind of an adversarial autoencoder.

So, we will be doing the same exercises, so, I will be showing you a bit of these examples over here with trying to generate mnist like images, and then based on whatever we learn and gather in our experience over here, I want to end up getting into more adversarial sample generation in the next lab classes, which will be recording the subsequent ones.

(Refer Slide Time: 01:52)


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

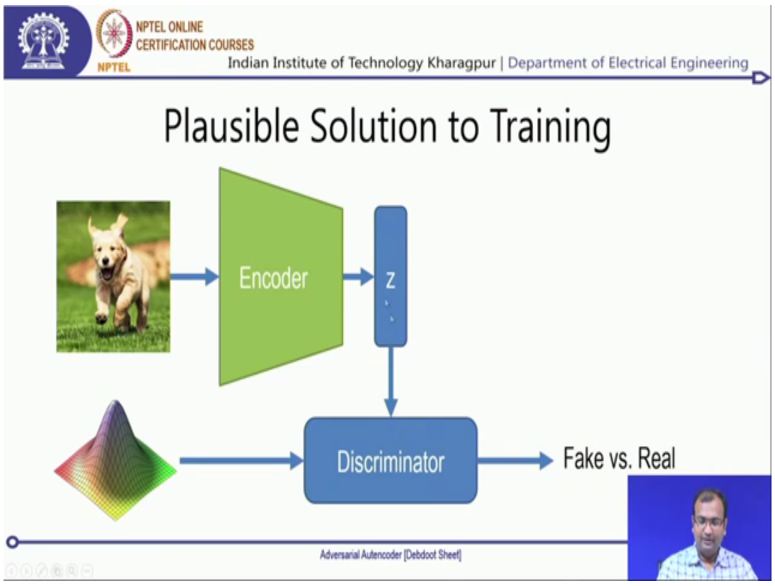
Plausible Solution to Training

Encoder
z
Decoder
Adversarial Autoencoder [Debdoot Sheet]

Now, one plausible solution to training which we had looked into on adversarial autoencoders was something of this sort. That you have your encoder decoder network present over here as in an autoencoder you put an image into it, and then you are going to get an image out over there. So, in within one Epoch if you train this over all the forward and backward passes, then based on that given one of these images from your validation set. You are going to get down this sort of a output coming down over here. Or from your test set from your training set also you get down a similar kind of image, but then whatever you find out between this image. And this image or the input and output error is going to be a nonzero value; this does not look exactly like the image which was given on the input. So, you have a non-0 error which comes over there and this non 0 error gets back propagated through this network end to end, and it updates out the weights and that is how this network is going to come down to a convergence.

Now, what we had on the other side of it? This was a forward pass training just for the encoder decoder to be able to act as an autoencoder.

(Refer Slide Time: 02:57)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Adversarial Autoencoder [Debdoot Sheet]

Plausible Solution to Training

Encoder
z
Discriminator
Fake vs. Real

On the other side of the pass what we had was, you generate a random sample from this 2 d Gaussian distribution, and then you have a discriminated network over there. And the whole job of this discriminated network is to find out whether the, this z or the latent variable which is a 2-d random number is coming from this distribution over here or was it coming from an original image. So, that is a real versus fake problem.

Now, well the discriminator is making itself stronger and stronger to discriminate whether it is coming from this part or this part, you also have a part of the error being transferred through your encoder, and that part of the error is a negative of the efforts of the discriminator. So, while the discriminator is trying to make itself very strong to find out whether it is from this source or this source, your encoder is going to make itself ah; so, there will be another update within this pass with it through your encoder which is going to make itself get down a value of z which is closer to this one.

And the very simple way of doing it is that, in case of you are calculating the error of this discriminator if you negate that error. So, if you make all the reels as false and all the fakes as real as fake and fake as real, then this encoder is trying to necessarily enforce that this z is something which should be looking like this,.

So, there is a twin game which plays around over here. So, in the first pass, I am trying to create an encoder decoder to create generate an image over there, and then in the second pass what I am trying to do is create a discriminator which can find out whether it is from a real image or it is from a fake distribution, and I will also update my encoder such that this z starts looking very much closer to this distribution. So, that is why taking just a inversion of these labels over here, on to that error and also back propagate. So, in my second phase of my training my encoder is also getting updated.

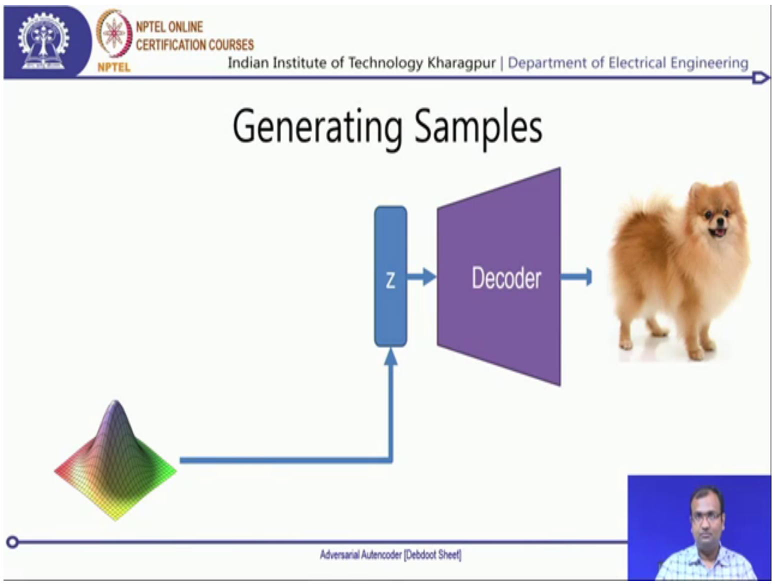
Now, once it gets updated in my first Epoch, I am going to start on my second Epoch. Now in the second Epoch, I am going to again have this encoder decoder structure coming down over here. Now in that encoder decoder structure now remember that what was the end state of this first pass of training in the earlier Epoch is no more valid, because in this second part over there, and again done a back propagation through this encoder and that had updated the weights of the encoder.

So now in this network, I have the weights of the encoder which is updated by the decoder weight is what is preserved from the previous Epoch over there. So, that will give me an erroneous result; and that means that this decoder will get updated as well as the encoder. So, this encoder is trying to force itself such that the z becomes closer to that 2 d Gaussian kind of a random number, and the decoder over here is always going to force itself such that, whatever is the 2-d random number which comes over here is able to generate an image which looks pretty much real.

Now, this whole kind of a game playing exercises, in which your discriminator is trying to make itself very good to discriminate real versus fake the encoder is trying to make itself very good to fool a discriminator, and generate a z which is close to this 2 d Gaussian over there, is as an end what you would get down is this encoder decoder network which comes out over there, and the decoder part specifically is something where provided you put a random number which is drawn from a 2 d Gaussian distribution, you will be able to always generate a real looking image. And that is the generating principle which comes down.

So, once you have your encoder decoder trained, you have a real image which goes down it will produce a real looking image on the decoder side of it.

(Refer Slide Time: 06:29)


NPTEL ONLINE CERTIFICATION COURSES

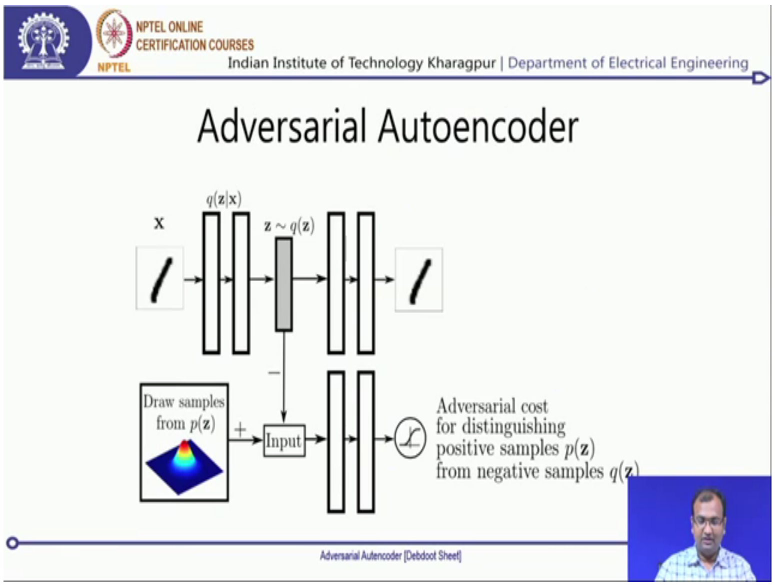
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Generating Samples

Adversarial Autoencoder [Debdoot Sheet]

Now, if I pull out my encoder and just draw a random number distribution from that and pass it through my decoder. I am again going to get down an image which looks pretty much real. So, that is the generating principle which comes out over there.

(Refer Slide Time: 06:44)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Adversarial Autoencoder

x
$q(\mathbf{z}|\mathbf{x})$
$\mathbf{z} \sim q(\mathbf{z})$
Draw samples from $p(\mathbf{z})$
+
−
Input
Adversarial cost for distinguishing positive samples $p(\mathbf{z})$ from negative samples $q(\mathbf{z})$

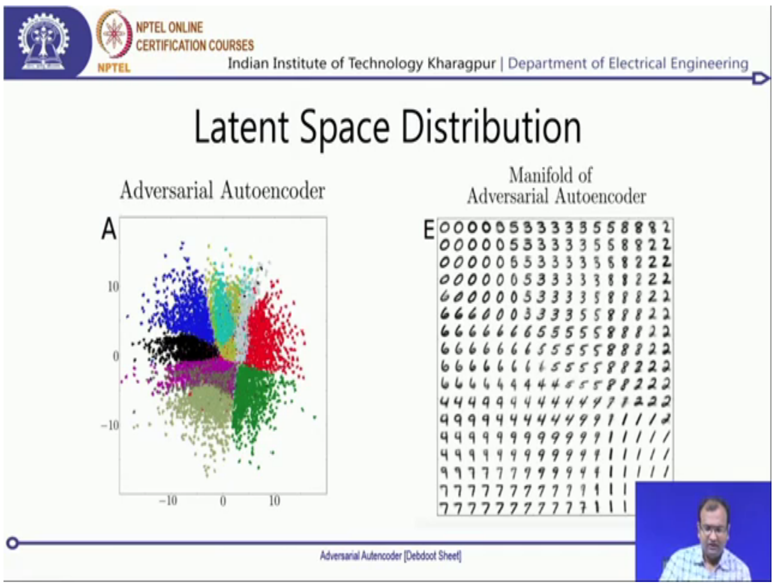
Adversarial Autoencoder [Debdoot Sheet]

Now, based on this there have been a lot of exercises which go down and the first and foremost of it is the standard architecture called as the adversarial autoencoder. So, this is taken down from the adversarial autoencoder people which I had. So, excited in the reading list in the earlier case as well so, the network over here is your input, which comes over here is something which looks like this also called as an x.

Now, this 2 layers from my encoder part of the network, this z is my latent variable and this part is my decoder part of the network. Now hi I train is quite straightforward. So, once I train this encoder decoder such that I get this output. Then I also train my discriminator over here. So, this is my discriminated network which comes here. Now that is supposed to find out whether the sample is coming from a random drawn distribution or from here.

Now, in a sense while my z is trying to follow down some distribution called as q z, but I am going to force this via adversarial learning that, this q z now starts mimicking p z over there. And that is when given any random number taken down from p z, because q z is going to be equal to or almost same as p z. So, if I draw a random number from p z, I am going to get down an image coming over here. So, this is a plain straightforward mechanism of doing an adversarial autoencoder.

(Refer Slide Time: 07:59)



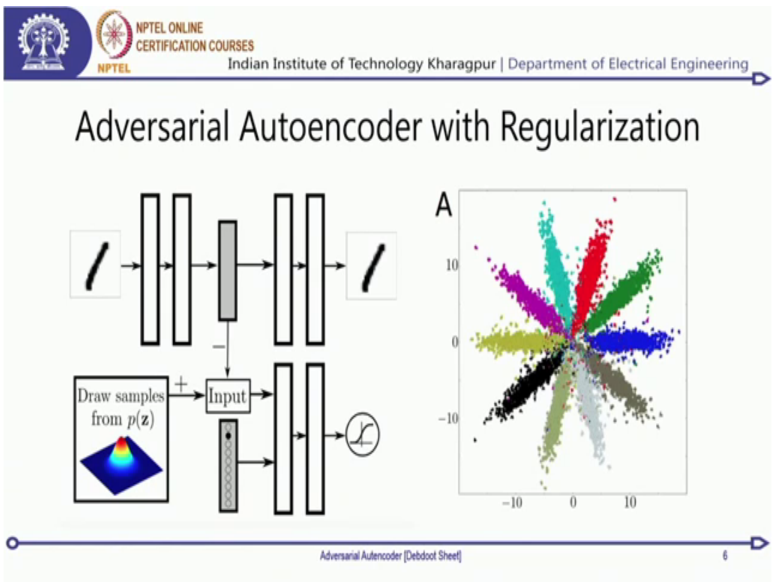
Now what we tried on the experiments was that we took mnist digit recognition point over there, and using this data set we were trying to. So, you just create an autoencoder and the adversarial part over there. And now look into this z so; this z for me is basically a 2 d latent representation. And now what we saw over there was that each of these colored clusters which you see, now these clusters belong to so, each color basically belongs to one class of a number so, 0 to 9 you have 10 classes. So, you see 10 distinct colors coming down over here.

Now, the z part over there does not in any way have this information on the class level or anything, they were just digits handwritten digits present over there. But then this cloud over here is something which looks quite similar to a Gaussian distribution, ok. Now some part of it is what belongs to each and every number and then that is how it is done now.

Now, if you look into this other side of it which is also called as the manifold for your adversarial autoencoder; what happens is, that if I am going to sample out from this region so, I have to 2 numbers, which varies from say somewhere around minus 15 so, minus 22 plus 20. Rather also minus 22 plus 20, if I start down some random sampling from these locations from here till here. So, I will add at some intervals fixed intervals I am going to do. So, if I do a fixed interval of 2. So, my first location over here comes down as minus 20 minus 20, then the next location over here comes down as say minus 18 minus 20, then like this it keeps on going.

So now for each of these random number if I feed forward that whole thing through my network so, I am going to get down and image being created, and this is just that matrix representation of the images. So, in the next lecture wherever you are going to record down and then show you exactly how this can be implemented on the code side of it. So, you will be seeing this kind of a matrix of random numbers being generated. Might not necessarily be exactly looking like this because you it depends heavily on whether random number whether initialization state of the neural network was and based on that you will get down what this appearance model keeps on changingly, ok.

(Refer Slide Time: 10:07)



Adversarial Autoencoder with Regularization

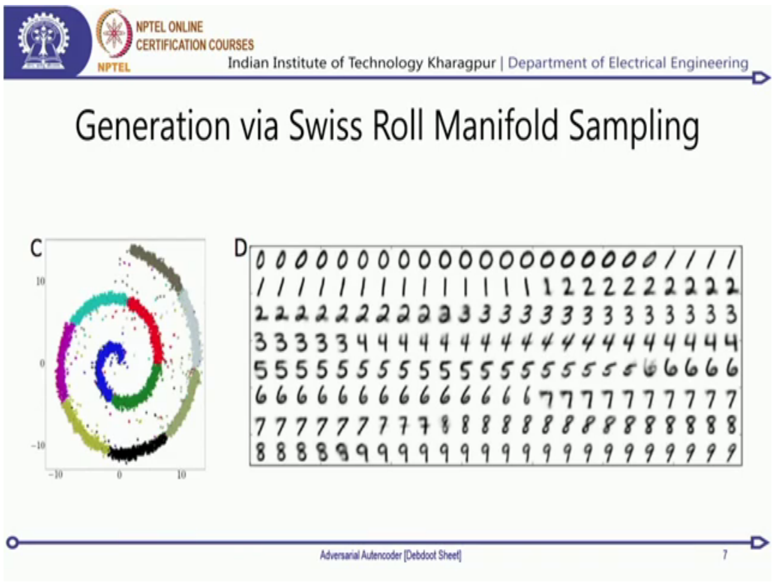
NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Draw samples from p(z)
Input
A
Adversarial Autoencoder [Debdoot Sheet]
6

Now there is another version of it, which is called as an adversarial autoencoder with regularization. So, what this does is another interesting aspect that you now start also including another variable which is called as the class variable over there. So, what I exactly want to do over here is that for each different class, I want to have a different kind of say random number being generated over there, ok, or something from a different distribution.

So, this p z which I am drawing over here is not necessarily from a uniform Gaussian distribution, but then this p z is from either one of them. So, whenever I am drawing from one particular cluster, I am also going to put down this extra information as to what is that cluster number. And each cluster is going to be one class itself over there. So, what this will help me in doing if we go back to our earlier example where did you wanted to have a model for generating cats, dogs, and everything separately, now I should be getting an option of specifying; like, whether I want to generate a cat or I want to generate a dog. So, that is the random numbers also get started pooling down from that distribution over there.

So, this is just extra information. So, what happens is you have a random number being drawn and also this class level information which gets passed on to this discriminator. Now nonetheless on your autoencoder part, when you have your encoder and decoder you do not need to pass this class information in any way that does not play any significant role over there. Now having trained this whole network in this mechanism what you can see is that if I try sampling out these random numbers following a Swiss roll distribution pattern.

(Refer Slide Time: 11:36)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
C
D
Adversarial Autoencoder [Debdoot Sheet]
7

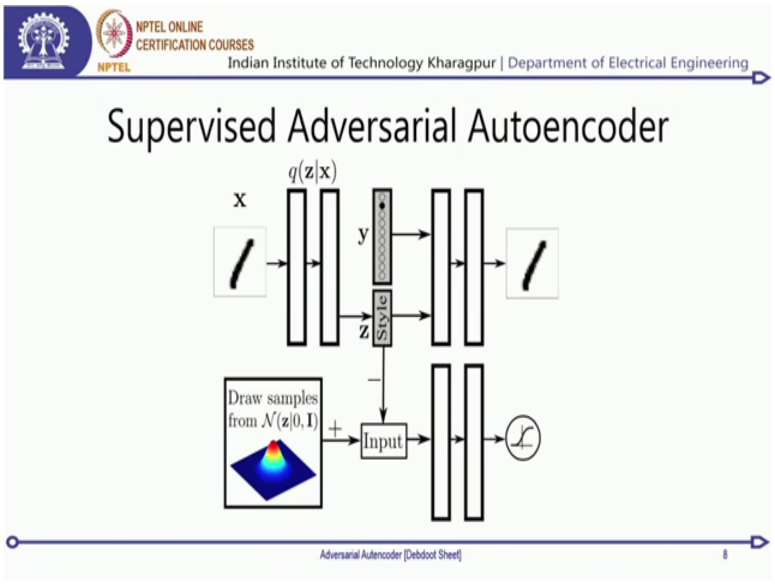
Generation via Swiss Roll Manifold Sampling

Then you see that there are different classes of numbers which come down, and then if you unpooled it out you would see these different numbers being generated.

Now, interestingly what you see is that you can actually choose, what is my number to be generated, and then it will be taking out a random value from that location to generate a number. So, that is that is the interesting part over that there.

Now, you have a control over the class of the model which you would like to generate or the class of the image, which you would like to generate which as such was not possible with the plain vanilla version of this adversarial autoencoder. So, this comes in much more handy over there.

(Refer Slide Time: 12:23)

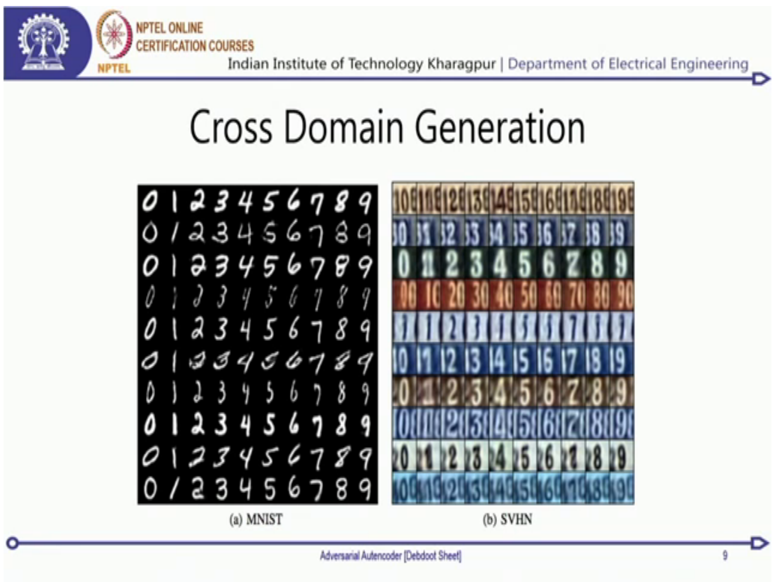


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Supervised Adversarial Autoencoder
$q(\mathbf{z}|\mathbf{x})$
x
y
z
Style
Draw samples from $\mathcal{N}(\mathbf{z}|0,\mathbf{I})$
+
−
Input
Adversarial Autoencoder [Debdoot Sheet]
8

Now there is also something which is called as a supervised adversarial autoencoder. So, in case of a supervised adversarial autoencoder where the difference goes is that this class level information is not given down to this discriminative, but that is something which is passed on to my decoder part over here, ah. So, it may be my class level information, or it may be another information from a possible generation model.

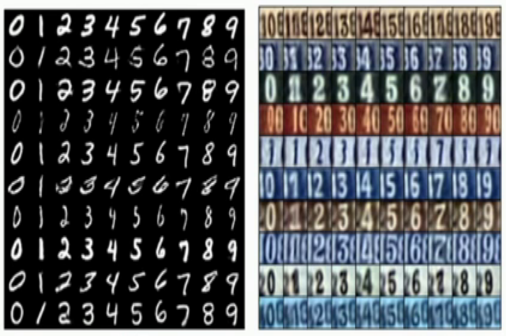
So, it is something like I want to have say sketches of dogs generated and not real like images of dogs. So, I will be training these networks in an adversarial mechanism, where whenever I am putting down real images of dogs I have one of these one hot vector set, when I am putting in a sketch images of dogs in order to simulate it I put down the second hot vector live over there. So, this way I can actually choose I still pick up a random value, but I choose what is my kind of a behavioral pattern over there, or maybe the context in which whether it is a sketch of a dog or it is a real photograph image of a dog to be taken down.

(Refer Slide Time: 13:30)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Cross Domain Generation

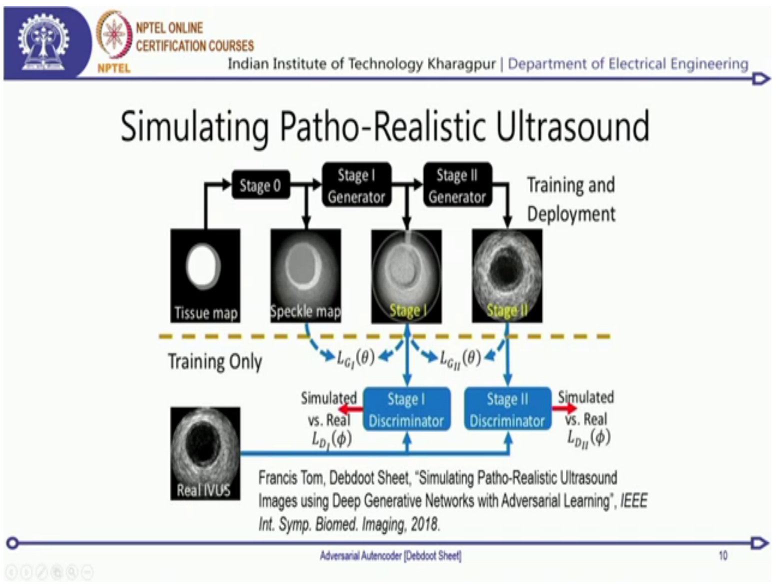

(a) MNIST (b) SVHN

Adversarial Autoencoder [Debdoot Sheet] 9

Now using that kind of an example, what we could see is that you can do a cross domain generation of that. So, here the example in this paper which they had taken was you could take these handwritten digits from say 0 to 9, and then you have this street view numbers taken down over that. Now make a marker point that these handwritten digits are actual handwritten digits on mnist, but then these numbers on street view house numbers is something which is generated. So, given one of these handwritten digits over that you would find out the z representation coming over here which is called as a style, and then you choose one of these hot representation. So, saying that I want to generate a street view house number based on this latent z representation, or I want to generates a handwritten number or maybe a printed character any of these.

Now, based on that I will be able to generate it out so, even if you do not have this house number present over there, now the advantage is that you can give the latent representation for that and corresponding to that there will be something on the street view house number equivalent which is getting generated so, this is cross domain generation. So, some other examples are like you give down sketch of images and you can generate paintings out of it. You give down sketches, and you can generate frontal face camera images or portrait images out over there. So, this has immense applications actually in order to find it out.

(Refer Slide Time: 14:50)


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Simulating Patho-Realistic Ultrasound

Stage 0
Stage I Generator
Stage II Generator
Training and Deployment
Tissue map
Speckle map
Stage I
Stage II
Training Only
$L_{G_I}(\theta)$
$L_{G_{II}}(\theta)$
Simulated vs. Real $L_{D_I}(\phi)$
Stage I Discriminator
Stage II Discriminator
Simulated vs. Real $L_{D_{II}}(\phi)$
Real IVUS

Francis Tom, Debdoot Sheet, "Simulating Patho-Realistic Ultrasound Images using Deep Generative Networks with Adversarial Learning", *IEEE Int. Symp. Biomed. Imaging, 2018.*

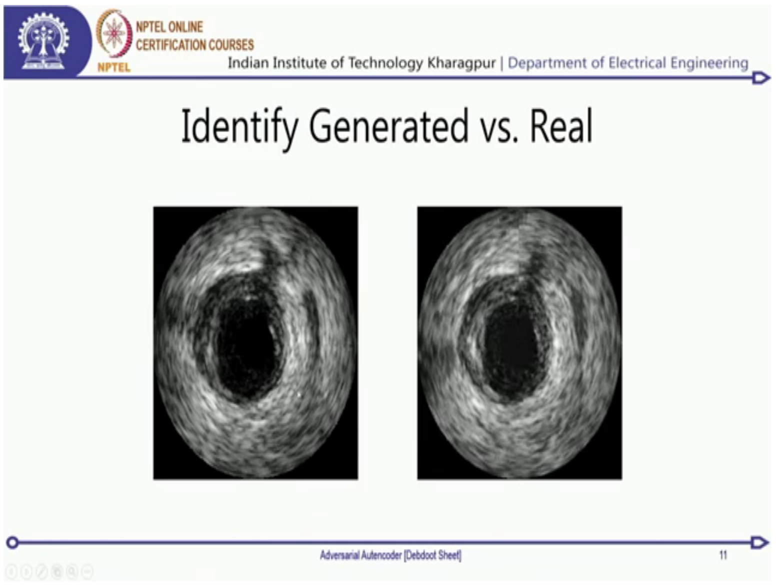
Adversarial Autoencoder [Debdoot Sheet]
10

Now one of these examples is from one of our prior work. So, this is work with we just have finished off an, and have it out for publication this year. So, the whole idea was we wanted to generate some of these kind of images which are ultrasound, and this is called as intravascular ultrasound images. So, these are these real ultrasound images which are present over there and we wanted to have a mechanism of simulating these ultrasound images. So, the idea was that if you have a phantom a tissue phantom over there which would more or less give you an anatomical structure, based on what tissues are present over there

Now, based on this anatomic can I be able to generate this one. So, that would mean that if I have a hand drawn sketch of certain anatomy is given, then can I generate an ultrasound image what will be corresponding to that. Now this would have good use in teaching in medical education. You can also have a lot of these synthetic images generated for really say strengthening your data set over there, and all of these purposes. So, this is what we had solved over here using a 2-stage generator. So, in which one of these stages was where you had to map out the objective was just to look into the average intensities and try to map it up, and the second stage was where we had to model out the point spread function of the PSF over then and then find out for.

So, though cost functions and everything were appropriately laid out. So, this has now been published out in one of these conferences, and you can have a look through the paper which is their own archive. Now if you look through how realistic it has been able to do the whole concept is still basically rooted on what we had done in the earlier cases.

(Refer Slide Time: 16:30)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

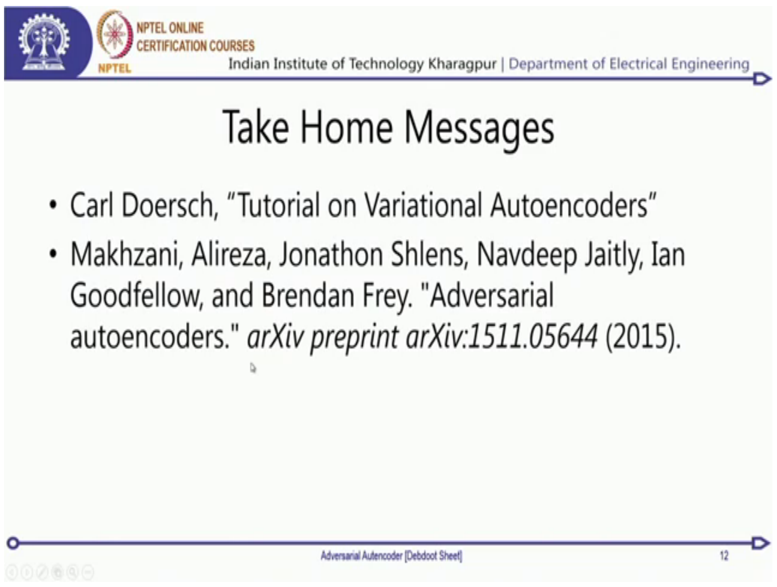
Identify Generated vs. Real

Adversarial Autoencoder [Debdoot Sheet] 11

And here this is a very typical example. So, if you look at the image on your left, then this is an image which is basically a simulated version, the image on the right is a real version. And now the anatomical map and everything was pretty much similar, and then it is it is really hard to find it or because there are these bright spots over here as well as over here which are certain anatomical markers. These dark spots over here also residue, residuals of shadowing due to a plaque formation within this hatchery, and they pretty much mimic each other to the closest form over there.

So, that is that is a really interesting problem over here, like while this one is generated this one is real it becomes really hard typically to do and over a experiment random for called as visual turing test in which you provide this pair of images to experts, who have been using these images and ask them identify which one is real. So, it is always a pair, and one is real and one is fake always, and you just your task is just to identify which one is real over there. So, this works are pretty good, because the results over there I just find 50 percent accurate, which meant that half of the time the fake image was always marked by this expert as a real image over there.

(Refer Slide Time: 17:44)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Take Home Messages

- Carl Doersch, "Tutorial on Variational Autoencoders"
- Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder [Debdoot Sheet] 12

So, that is where we come to an end with this one. And so, you have your standard take home message which has these papers to refer down to and then this adversarial autoencoder paper is really nice to go around with, if you want to look more into advanced parts of these modelling, then you can have a look into the earlier referred paper on this intravascular ultrasound generation from us as well.

So, with that we come to an end with trying to understand the basic concepts of adversarial autoencoder in the next class we will be going through the lab session on how to create an adversarial autoencoder, and how the generative principle over there will for. So, with then we come an end to this lecture and stay tuned for the next ones to come up.