

Deep Learning for Visual Computing
Prof. Debdoot Sheet
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 51
Autoencoders and Latent Spaces



Welcome to this week series. And then this is where we get into a real interesting aspect about deep neural networks and might have heard about some of these things making a real buzzword called as GANs or generative adversarial networks. Now, where it comes down is this interesting aspect about generation ok, and that has a lot of its basis quite rooted into our concepts of autoencoders.

Now, that is where the title for today's talk is given down as autoencoders and latent spaces. And we try to understand; what is the intermingling dependency between autoencoders and latent spaces and that will lay down a very good foundation into our understandings of how a generative model actually works out ok.

Now, I would be revising a bit of the concepts of auto encoders in terms of trying to again do the linear algebra and the math around with an autoencoder, and then get into the concept of what is called as stacking of trained coders. And from there we will now so you had already done these stacked auto encoders. And there were two different stacking strategies which we had studied one of the stacking strategies was where you had a layer wise pre training, so that was a ladder kind of architecture in which you have one hidden layer at a time which is added down to your network. And as you go along the depth and that is also increasing your performance over there as we say.

Now, there was another mechanism in which what you could do is you could train the encoder part and train the decoder part together, and then chop of the decoder part and then just place down your final classification layer over there. Now, this is the kind of a network which we are going to make use of a fixed end coder and a fixed decoder, and how it goes around with that one. So, without much of a delay let us get into exactly what I am going to speak on with that.

(Refer Slide Time: 02:07)



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

Contents

- Revision of Concept of Autoencoders
- Latent space in AE
- Statistics of the Latent Space
- Generative Principle

Autoencoders and Latent Spaces [Dehdoo Sheet] 2


So, I will be briefly revising the concept of auto encoders and then entering into what is the definition of a latent space. Now, I have been speaking about this latent variable or lot of times and if you recall from your autoencoder lectures, and there was a variable called as z which was called as a latent variable. And this latent variable is what was being referred incrementally over there. So, for each of these hidden layers over there whatever was the output of the hidden layer is what is called as a latent variable.

So, you have input from the previous layer and then your basic multiplications and then convolutions or full connections over here was a tensor multiplication with the weights present in that particular layer. And then you had the output coming out of that layer and the output from this layer is what is called as the latent space. Now, here we are going to revise into what is called as the latent space of autoencoder and what is the distribution of the statistics around those latent space.

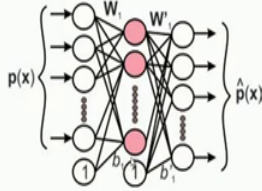
Now, once you enter into the statistics that is where the interesting parts starts coming into it, and then I enter into the generative principle. So, this is where it we come to an end today in terms of just understanding where the generative principle would go. And then in the next lecture, I would be coming down into the math part of how to impose this statistics over there, and how this generative principle is actually going to work for each of them. So, we will start initially with very simple mechanism of generation of

different examples and then from there onto images and how it goes and so on and so forth.

(Refer Slide Time: 03:33)



Autoencoder



$$y = [w \ b]. [p ; 1]$$

$$\hat{p} = [w' \ b']. [y ; 1]$$

$$J(W) = \sum_n \|p_n - \hat{p}_n\|$$

$$\{w, b, w', b'\} = \arg \min_{\{w, b, w', b'\}} (J(W))$$

Autoencoders and Latent Spaces [Lecture Sheet] 3

So, now if you get back into your basic understanding of an autoencoder, so what it said was that say I have coordinate location x and a small patch p which is centered at that coordinate location x . If I take each of these pixels from this patch, so that is going to consist of 1, 1 neuron over there and the input to this neuron is what gets into it. Now, as it gets into this one then you have your weighted connection and you have your intermediate hidden layer representation over there.

Now, on the autoencoder part, what happens is that whatever comes out of this hidden layer is what we also call it as a latent space. Now that gets again fully connected and then you have this reconstruction happening over here. Now, for each of these times you have the bias, which is also added down, and this is independent over there. So, the bias the except for like really getting into the standards of how this optimizations also over there, how your error is coming down and the learning rules over there, this bias does not have any other significant role. So, it does not skew you down the nature of the latent space, neither does it even in anyway hampered down the nature of the input over there.

Now, how it went was quite clear that if you have input p over here which is given down to these neurons. Then this is this y is what would look as the output from this hidden layer; and provided that you do not have some sort of a non-linearity over here and that

is what we are imposing over here. So, there is no tan hyperbolic or say sigmoid kind of a non-linearity given down, it is just a plain simple linear equation. So, you have just a tensor product between two tensors coming into it.

Now, from here you would be able to reconstruct your output patch and that is \hat{p} your reconstructed version. So, this is also a simple tensor product which comes out over here. Now, what we said is that we would be taking into account something called as a cost function. So, this cost function is what is called as J of W and this was just my Euclidean distance and. The whole idea was that I would like to minimize this Euclidean distance over all the possible samples n . So, if I have some n number of images which come on my input over here, I will be able to get one, one corresponding reconstructed image over there.

Then over all the images present over there, I am going to take a sum total of what is this absolute value of difference. And this absolute sum of absolute value of these differences over all the n images present in my corpus should actually be as minimum as possible. And now since you have this Euclidean distance being coming into place. So, the Euclidean distance always being a positive number. So, this sum if this has to come down to a minimum that can come down to a 0 and nothing else. And what that would essentially also impose is that since each is a positive number a number which can be just greater than 0, but never a negative.

So, this series summation will only be 0 only if individual component over there is 0. If any one of these images has a very high error, it will still not be coming down to the desirable quantity over there, so that was my way of actually establishing whether my network is properly done and then in order to get down my weights which are my only free parameters over there. So, my w_s and my b_s which are biases; and w_{prime} which are the weights which connect on hidden layer to the output. And b_{dashed} which is my connection between the bias to the output over there.

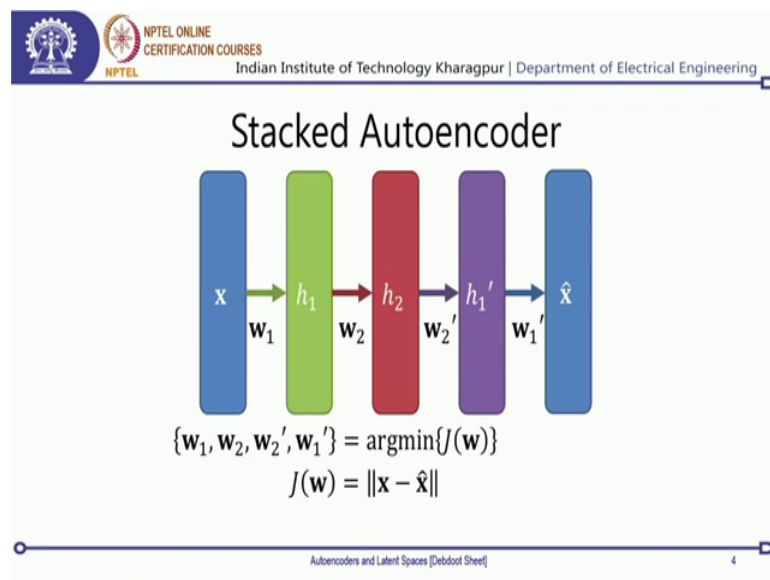
Now, this thing in order for this whole thing to be optimized over there, I need to minimize this cost function. And the whole idea behind minimizing this cost function was that we employ some kind of a mechanism in order to minimize the cost function. And as we are able to employ this mechanism to minimize the cost function, we would be having a network where given an image on the input side, now you can get the same

kind of an image on the output side. And in the whole process, we are basically learning of these features which represent the image in terms of these fits w s ok. Now, this was one part of it.

Then you had learnt out another interesting use case for these kind of a network and that was with a denoising auto encoder. So, the whole idea was that I give a noisy image to the input and try to enforce the output to be as close as possible to a clear image. And the major advantage for this kind of a network is that you can start denoising your images, you can do a noise removal of your images using this kind of a network; now that is well and good and a pretty standard which works out. Now, the point is it possible in some way to generate images as well and that is the question for any kind of a generative model.

So, if I have these intermediate hidden layers outputs, somehow fed down over here, then can I still end up generating images, so that is that is what we are going to solve out in today's example.

(Refer Slide Time: 08:17)

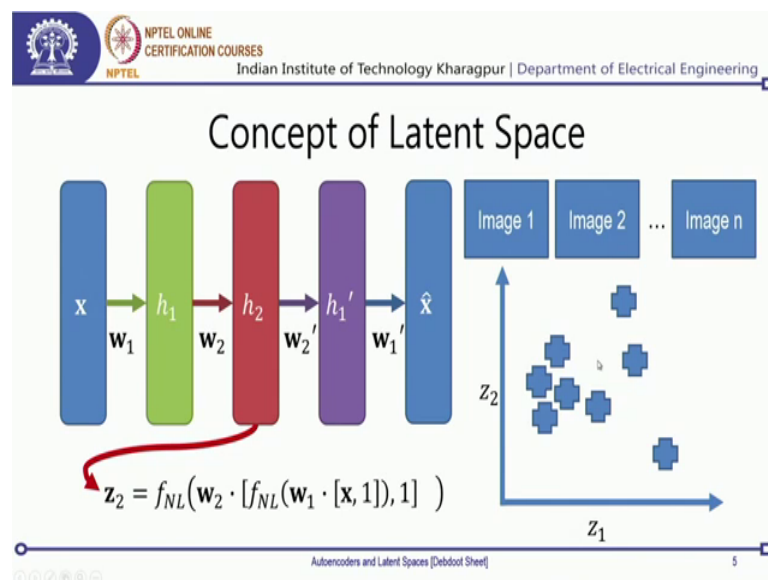


So, let us take this standard example of an stacked autoencoder. So, what you basically had was given an input x so this was my image, you have one hidden layer via your weight w_1 then you had your second hidden layer your way w_2 . And then the output of the second hidden layer is what is called as your latent variable z ok.

Now, on the this part over here is what constitutes your encoder, then the next part over here where you are going to reconstruct h_1 dashed from h_2 is where your decoder starts. And now from h_1 dashed you are going to recover your \hat{x} . So, \hat{x} is basically the recovered of the deconstructed version over there, and this blocks in violet and blue are the ones which constitute your decoder. And what we have in this whole process is that this can be solved out only if you have this kind of a arg minimization principle taking place.

So, this was my straightforward mechanism for a autoencoder with multiple layers. Now, in the earlier case you had seen down only autoencoder with just one hidden layer and this is you have an autoencoder with multiple hidden layers.

(Refer Slide Time: 09:24)



Now, what comes out is quite interesting. So, if I have say only this part of it where I have an input and it goes down to my first hidden layer then that goes down to your second hidden layer. And now what I do is I can keep on reconstructing this whole thing, but now I look at tapping out this intermediate from my second hidden layer. This, so this tapped out formed from my second hidden layer; so the output not the weights or anything the output given an input image x , what is my output which is coming from my second hidden layer.

So, the output from the second hidden layer is what is known as z_2 . And if I have a non-linearity in place then this is a kind of an equation which describes. So, the first point is a

tensor product between your input and the weight, weight down by your non-linearity, and then you have a tensor product of this whole thing with the second layers weight over there, and this will be weighted by your non-linearity over here. So, this is my output which comes down from h_2 and then known as it.

Now, as this keeps on going over here so let us have a look at what will happen on an image-to-image basis over here and that is to look into how will this z be impacted. Now, if I assume that this h_2 just has two neurons so it means my z over here basically has two dimensions. So, if h_2 is to by z is two-dimensional thing. And now it becomes easier. So, let us let us try to visualize it on a two-dimensional plane of picture.

So, if I have z which is broken down into two component axis z_1 and z_2 ok, now for one given image, I will get down a point over there for my second image I get down a point and so on and so forth it keeps on going for all the other images. So, for every image, I am going to get a point in this coordinate space of z_1 and z_2 available over here.

Now, as I keep on going down till y nth image, I will be getting down this point located corresponding to my n th image. Now, essentially if you look back into this concept, then what this would mean is that for every single image I can encode via this pipeline and get a two-dimensional representation if I have a two-dimensional hidden latent variable over here. In case I do not have a two-dimensional latent variable, then whatever is the dimensionality of written variable on that space I am going to get a point. So, every image now basically boils down to one single point in my space, now that works out pretty fine.

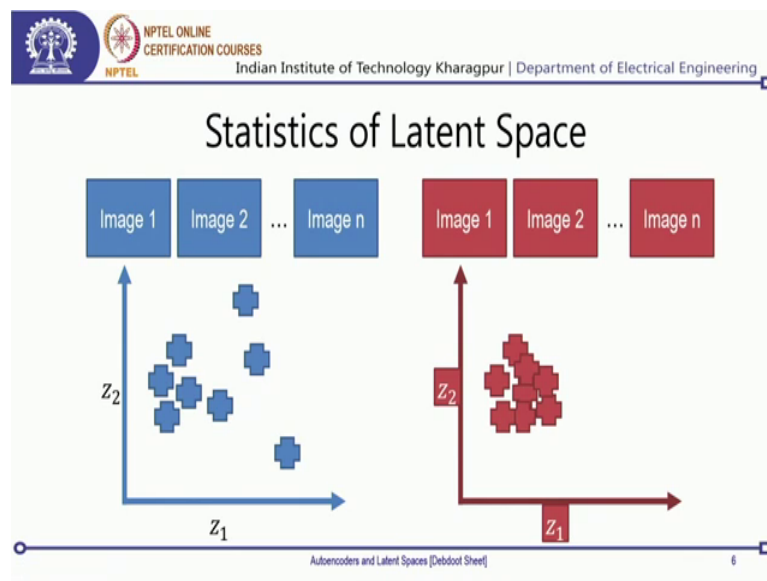
Now, if that is the case what I can essentially do is I can pull out a point from this space and feed it to this decoder, and then I will be able to even reconstruct my image that should be that straightforward. Now, apparently what turns out is that this space as such on which this falls down is something which is really hard to sample because you can pretty much see that these are scattered out over here. So, for any kind of a practical example also it gets pretty much scattered around over there. So, it is not something that you can easily pull out a valid representation.

Now, if you get back onto this space to look into it what they would see is that you had this say you had this point which was corresponding to image one. Now, if I point and

then you had this other point which is over here corresponding to say image n. Now, if I pull out any random point which is located between these two locations, my assumption would be it would be giving an image. But then, it is not necessary that it would always give you a meaningful image which meant that something which looked like an image I mean it might be something which is totally unrealistic. So, say a sun in the middle of the ocean or say a moon within a sun or a cat with eight legs. So, it can even end up with those kinds of weird looking manifestations over there.

Now, the whole point is that for any kind of this kind of a network wherever you can give a random number, it is going to generate you an image at the end of the day. The only point is whether the image has certain meaning and sense to carried on and whether it looks like a photorealistic or a real image or not is definitely a question which we need to solve over there. So, this whole aspect of adversarial modeling and generative principle is to look into whether whatever is generated on the output side, whether that makes sense and it is good enough to be as if you vocal as a real image.

(Refer Slide Time: 13:39)



. So, that is where the statistics of latent space really comes into play. So, say I have the same kind of an example to access over here there is two dimensions on my latent space. And then for every image I am going to get down a coordinate point over there. Now, it is it is great you know now this comes down from one of my networks where say all my points that my blue. So, I can call this as a blue network.

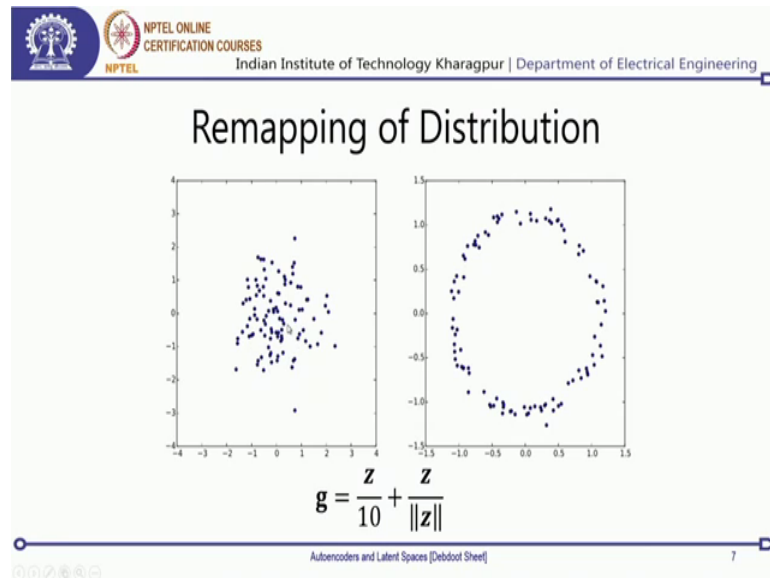
Now, let me create a separate other network with initial random selections of weights over there ok. Now, if I create another network which I call as red network over that. So, I will have my $z_1 z_2$ distributed something of this sort. Now, for every single image, it is going to give me another kind of a distribution.

Now, if you look into this kind of a form, what you would see is that the distribution of these latent points in this second network is something which is more coherent. And there is a high chance that there is some valid point which is present between two different points which you have created over there or in a sense that these latent representations tend to cluster. And more generally what you can also say is that the density is somewhat concentrated around this point which you can call as the cluster centroid to a certain extent.

Whereas, over here in the first case of the blue network, you do not see such kind of a clustering; and as a result what happens is if you are trying to extrapolate it out say if these are the boundaries of my points present over here, these are the boundaries of my distribution. Then I can say that maybe there is a point valid in between which would also give me a valid image now that is not always true. Whereas, over here what you can see is that this has a much more cohort and a concise boundary. This actually forms down a much clustered convex representation; whereas, this one does not fond. So, this has some sort of a non convex representation which comes out.

So, most generally you would see that all the valid points are something which exists over here and may not be over here. So, if you are going from this point to this point, you are just outside the set. And that just violates the whole principle of being able to generate any particular image given down a particular given that you take a random value as the input over there.

(Refer Slide Time: 15:50)



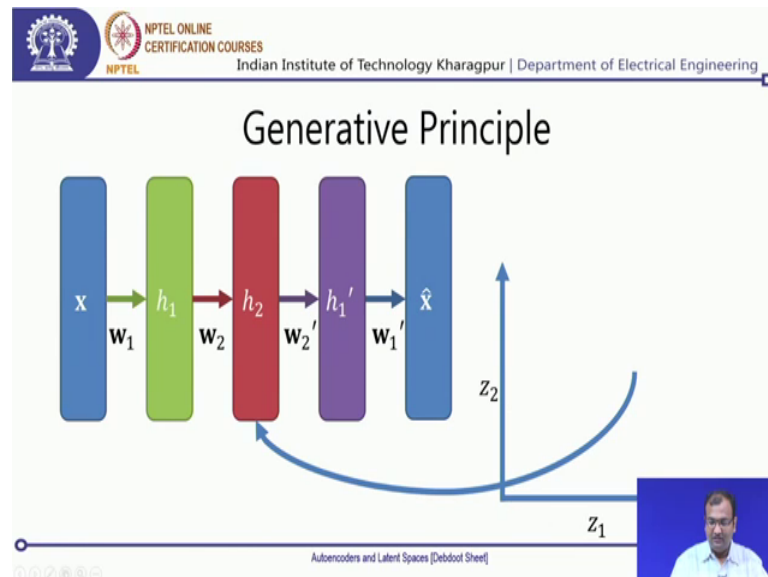
Now doing that where it comes off interesting point is that I can have these kind of clustered points over here. So, this is taken down from a very classic example. Now, in the earlier case what we said is that if your points are more or less scattered around over there, then it becomes problematic to a really find it or because you will have to choose from these points only in order to generate anyways. If you choose you choose a value of z_1 and z_2 from any other location, so if I am choosing it from somewhere in between over here and not exactly from this periphery this ring like periphery of points which are scattered over here. Then there is a high chance that I will get down an error and some weird looking thing which is not even close to an natural image over there.

Now, on the other side, I have this close cohort of clusters over here which is supposed to work out pretty fine. Now, there is actually a very easy way of transforming any kind of this kind of a cohort of clusters onto these ring like arrangements over there or a non convex. As well as any kind of a non convex arrangement can not necessarily always any kind of a, but to a certain extent a lot of these kind of non convex arrangements can actually be made to formed on these kind of clusters over here.

So, as an example say if I have this kind of a transformation where my z is this variable over here, and my g is this variable over here then I can relate down any point on this space to a point over here and vice versa. So, this makes it really interesting that at certain kind of geometric transformations, you can actually map down these kind of

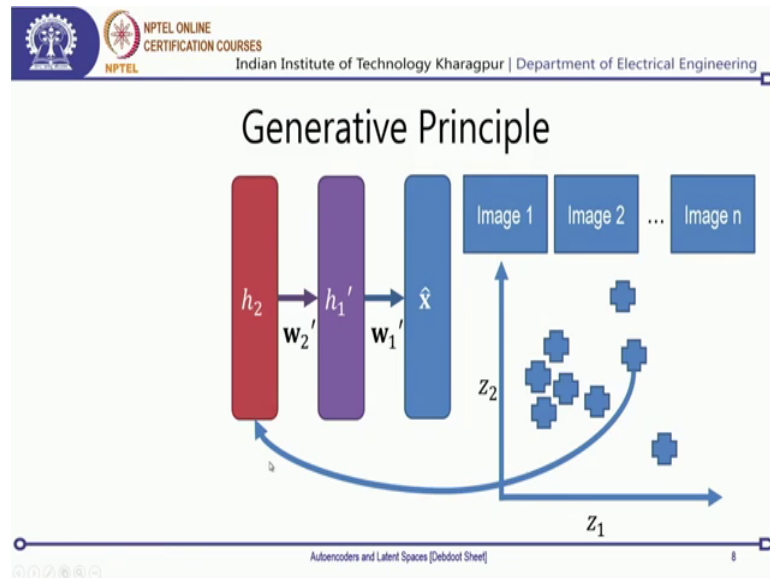
scattered around points to form a very convex set. And this has a very important implication because the geometric transformations can also be modeled down with a neural network itself and that comes as one of the major saviors over here.

(Refer Slide Time: 17:38)



. So, as a result what would happen is if I am looking at a generative principle, then what comes out is that in general you had these encoder part over there. And now you are using only this encoder part in order to solve your problems. Now, what I would do is let us get down this decoder completed. So, you have your encoder and decoder taken down. Now, if I have this kind of a 2D sampling out space, and from there I decide that I will just neglect the decoder part and have a sorry neglect the encoder part, and just have this decoder part.

(Refer Slide Time: 18:11)



So, now what I can do is I take a random sample from any of these points over here, then I will be able to generate any kind of an image over there so that is the whole aspect over here which comes into my generative principle. So, what essentially you do is you are trying to build an encoder decoder network in a way such that these latent variables over there follows some sort of a distribution. Now, what that would allow is that you can randomly pluck up any number from a distribution it is easier.

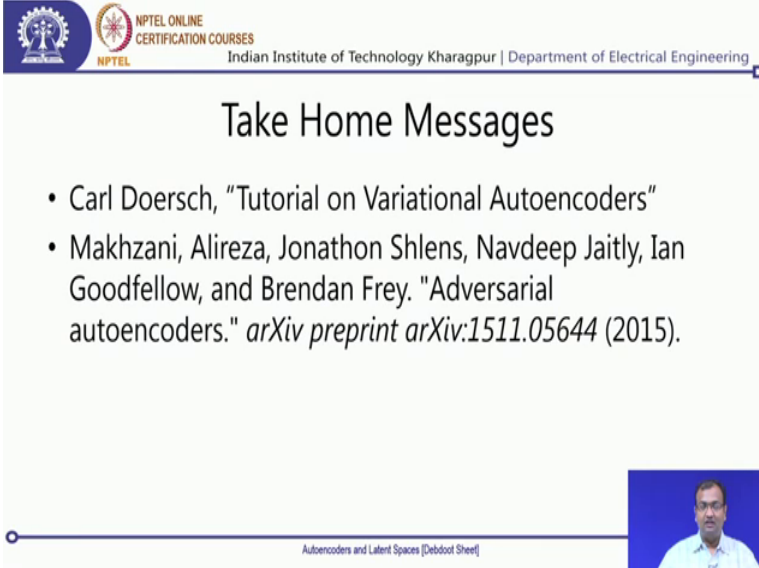
Now, every single image on your training set would be mapping down to one unique point on your latent space. And now if I am picking up exactly that point, I am going to reconstruct this image, but if I pick up a point within that cluster which is not exactly a point which was present on my training set, then I am going to pick up points which are quite different. Now, whatever images will be created out over here on the other side of it, we will have newer kind of a manifestation. So, they would not look like exactly the same image which was there in your training set maybe they will have similar content, they will have similar look and feel over there, but there will be certain subtle variations. And what this would allow me to do is on a bigger side generate a lot and lot more of synthetic datasets coming down.

As well as now one you can do is you can generate of synthetic datasets use these synthetic datasets in order to train your rest of the computer vision algorithms. But look at the other side of it if a model is able to generate these synthetic datasets very easily by

pulling a random number from a distribution, then the encoder and decoder and the features which these models have learnt in the encoder part as well as in the decoder part, they are robust enough for any kind of a sample which it has not seen or what can also be called as an adversarial sample. A sample which was not seen by the network during the training and it can still identify this one. So, the generalizability of these networks which starts becoming much higher and higher.

So, this is the whole concept of generative adversarial learning or generally kind models with adversarial learning and there is a difference between when it is called as an adversarial model, when it is not an adversarial learning model, so that there is a learning paradigm which comes into play. And this we will be doing in the subsequent lectures. But overall the idea is that if you are able to create a network where by pulling in these latent variables ran as a random number from a particular distribution, you are able to generate and synthesize an image then that works out pretty fine. So, that is the whole concept of a generative modeling.

(Refer Slide Time: 20:40)



The slide is a presentation slide from NPTEL Online Certification Courses. It features a header with the NPTEL logo and text: 'NPTEL ONLINE CERTIFICATION COURSES Indian Institute of Technology Kharagpur | Department of Electrical Engineering'. The main title is 'Take Home Messages'. Below the title, there is a bulleted list of references. At the bottom right, there is a small video inset showing a man speaking. At the bottom center, there is a footer text: 'Autoencoders and Latent Spaces [Debdoot Sheel]'.

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).

Autoencoders and Latent Spaces [Debdoot Sheel]

So, now finally, if you want to read more about it, then the best place to go around is this Tutorial on Variational Autoencoders by Carl Doersch. And also you need to go through this particular one on adversarial autoencoders by good fellow and his team over there. Now, this particular paper on adversarial auto encoders is what we will be studying in detail as well and as one of the best treaties on this particular field.

So, till then stay tuned. And we come back in the next lecture with more amount of the mathematical fundamentals of how to get down these adversarial and generative models working down.

So, till then thanks.