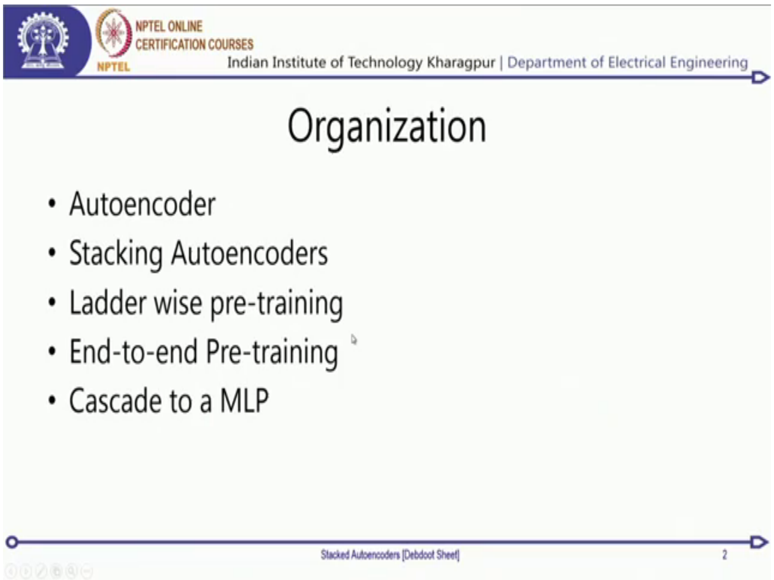
Deep Learning for Visual Computing
Prof. Debdoot Sheet
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 16
Stacked Autoencoders

So, welcome to today's lecture. So, till yesterday we had done on Autoencoders and then subsequent to that we did have a lot of exercises and they were very basic exercises on trying to deal with number of images and of different varieties and then you also saw down how to do a pixel to pixel classification, how to do a patch to patch classification, how to handle down gray scale images, which are just in one channel; versus how to handle down color images and over multiple classes. So, ranging from just an 10 class classification problem to multiple class classification problems as well as we even read down one particular one where we were trained to do patch to patch paced pixel to pixel classification over there.

Now, that was just doing simple Autoencoder today what we would be doing is using down something called as a stacked Autoencoder. So, a family of stacked Autoencoder is something which if you carefully look into it. So, this is a way of basically stacking down your Autoencoders or how to do it basically by stacking one layer after the other and as I was telling you that when you have these hidden layers over there one by one in a multilayer perceptrons. So, one of the challenges is that the dynamics of the total network depends on dynamics of each layer itself and then if we have some way in which we can actually freeze down one layer at a time and then keep on training it subsequently it would be much easier to train down the complete network.

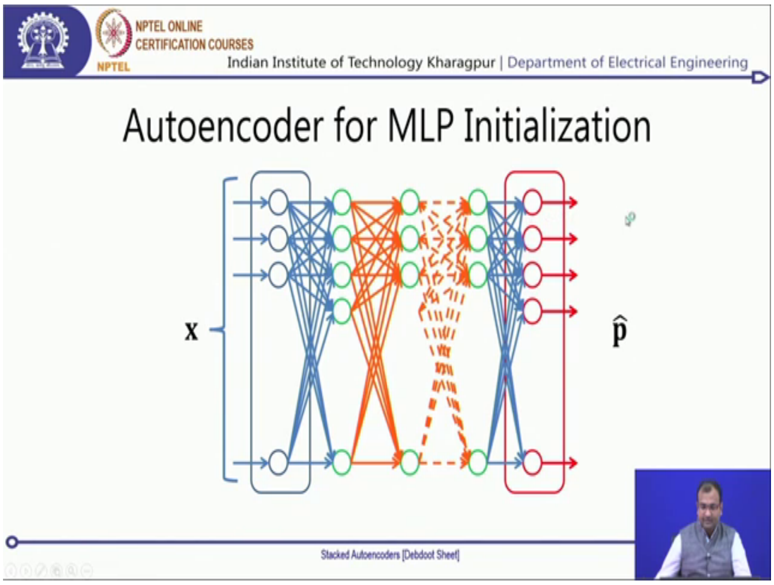
(Refer Slide Time: 01:43)



So, without much of a delay let us get into what it is. So, I would be introducing again once again the basic concept of Autoencoder and, but you knew about it then I would be speaking about is called as stacking Autoencoder. Then something called as a ladder wise pre training. So, there are 2 basic options of doing a Autoencoder training basically one method is what is called as a ladder wise pre training and; that means, that you grow one hidden layer at a time.

Now other method is basically using something called as a end to end pre training which means that you create the Autoencoder structure over there and train it in one single go. Then you can cascade like break up part the Autoencoder part over there have down all the feature representations and then try to cascade that with multilayer perceptron in order to with just sigmoid layer on the decision side in order to create something called as a multilayer perceptron.

(Refer Slide Time: 02:36)

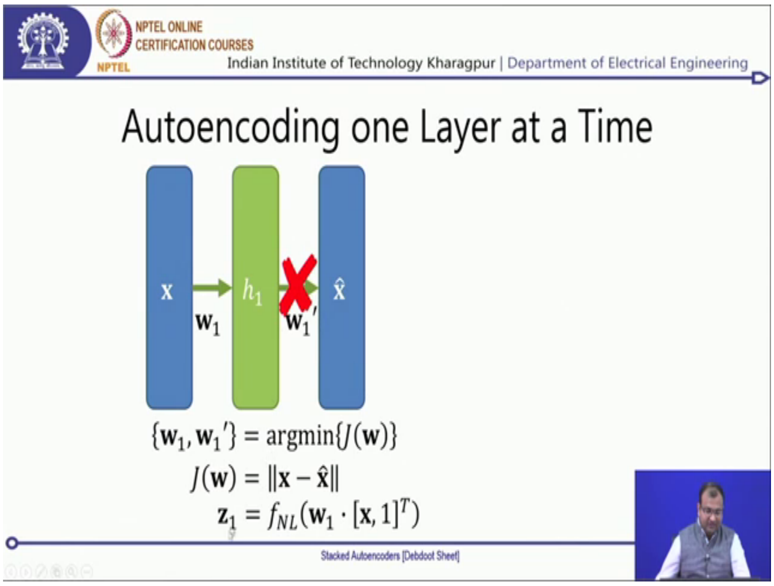


So, again revising it down what happened in a Autoencoder was something of this sort that you had your input x. Now if this is an image then this is basically set of all the pixels present in the patch of that image. And now using all the pixels over there you can basically arrange all the pixels into one single one neuron over there. So, if I have a 5 cross 5 Patch it mean Andit's a gray scale image. So, it means that there are 25 pixels over there. So, I will have 25 such inputs neurons over here now if this 5 cross 5 patch, but from a color image and RGB color image. So, it means that there are 3 into 5 into 5 which mean 75 pixels over there.

So, would be having 75 nodes in this input layer over there from there I connected down to the first hidden layer. Now this first hidden layer will have some n number of nodes over there and all of these neurons and input will be connected to all the neurons on the output. Now again from this first hidden layer to the second hidden layer everything will be connected and that is how your standard MLP is formed down, but what we want to discuss out is if this a pure Autoencoder, which means that p hat whatever is predicted is equal to x that is what we would like to do. Then how can you use that in to and how can we have different ways of doing that one.

So, one technique or as I call basically there are 2 techniques. So, we will start with calling one of them referring one of them as one technique or ladder wise pre training technique and the other one is which is an end to end learning or the other technique over there.

(Refer Slide Time: 04:09)



So, here the idea is basically that you would like to autoencode one layer at a time. So, while trying to do an autoencoding of one layer at a time what we would typically start with a say I have an input x and I have 1 hidden layer h 1, and then whatever I am predicting out is what is called as x hat and that x hat is something which is supposed to be similar in it is all forms to x as well.

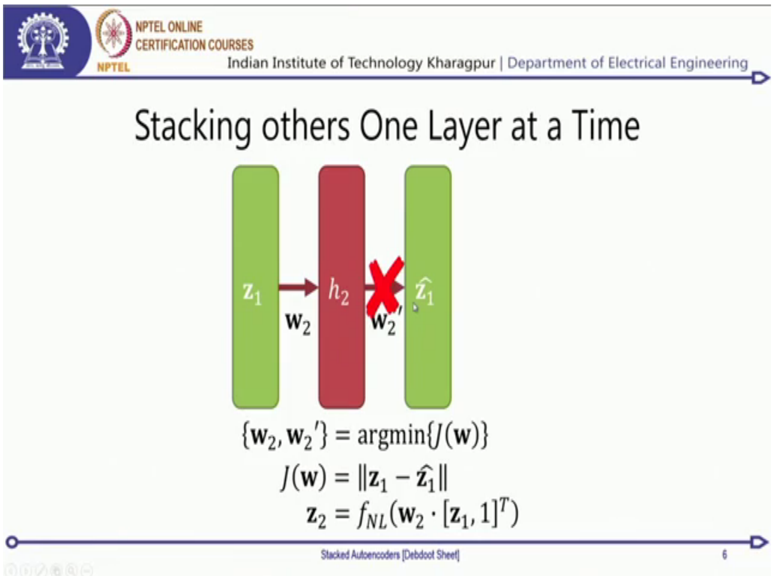
Now the weights which connect down x to h 1 is what is called as w 1 the weights, which connect h 1 to x is what is called as w 1 dashed and we did do from last week's simple mathematics is that this h 1 is completely dependent on w 1 and x hat is what is dependent on w 1 dashed and the reason we put down dashed is because this is what is trying to symmetrical relate to the other side of it.

Now, when you have this sort of a form over there so what you can do is while training this is the basic training algorithm which I am going to use over there. So, my objective is that I would like to minimize the cost function for all of these w 1 and w 1 dashed such that like wherever w for w 1 and for whatever value of w 1 and w 1 dashed I have my minimum error coming over here, that is best composition of these 2 weights and this error is defined as the Euclidean norm or the l 2 norm of the input and the output over there and this means that my best case is when x is equal to x hat and that is when my error will be 0 and that is the best form of an Autoencoding.

Now once I have trained this one I would like to include another hidden layer, but what I do in that case is after this training is done I would be chopping off this weights over here and if, I chop of this weights over here I can just look into my outputs from h 1 and the output for my h 1 is something which can be defined like this as z 1. So, output of my first hidden layer is what I call as z 1 if you carefully note down that I have written down these tensors over here in the form of bolt. So, these are basically arranged. So, some matrix form of an ordering of scalar values which is present over there and then my non-linear function over there will do it over this tensor in a product.

Now, once I have these a outputs from here which is at one what I can do is I can look into stacking the other layer.

(Refer Slide Time: 06:28)



So, in order to stack the other layer what I would do is I would no more be taking my input x over there, but once that network is trained one by training data what I can do is I can use all of my training data and transfer it through that network and get an equivocal representation of z 1.

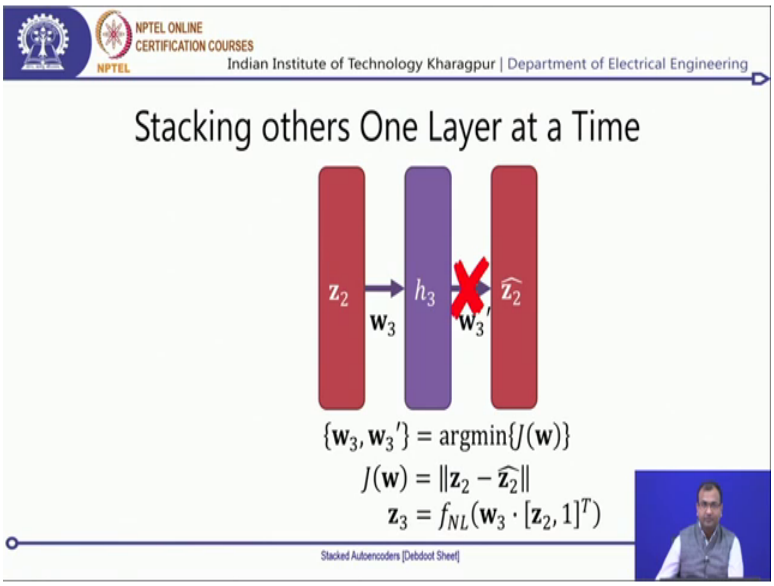
So, for my first sample of x I have will have one equivalent sample of z 1 for my second sample of x, I will have an another equivalent sample of z 1, then that is how this whole training set can be transformed into another transformed form and that is known as z 1 transformed form. Now here for training it what I would do is that I would use this z 1 and then try to reconstruct z 1 itself and that is z 1 hat and the 2 weights which will be associated now to the second hidden layer is w 2 and w 2 dashed. So, you use that same sort of an argument over there and you would try to minimize this difference between z 1 and z 1 dashed and if this one comes down to 0; that means, that you are at the best possible combination of w 2 and w 2 dashed.

Now once that is done now I can chop off my weight layer over here w 2 dashed. Now I chop of my weight here on w 2 dashed what I get is a second sort of a latent out port which is called as z 2. So, z 2 is a transformed version of z 1 which I have over here. Now for once this training is done then what I can do is for each value of z 1 I can transfer it through the set of equations and get down z 2. So, it means till this point I have each single value of x on my training set represented in a form of value of z 1 and also represented in another form of values for z 2.

So, for each input patch which I give I have a set of features for that patch in terms of z 1 I have another set of derived features which come down from z 1 for each of these patch and that is my z 2.

Now once this part is done.

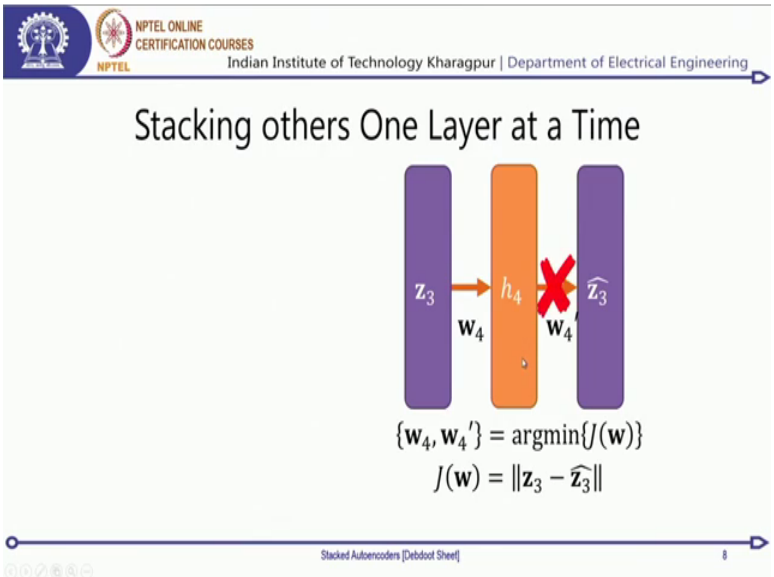
(Refer Slide Time: 08:26)



Now, I can stack another layer on top of it and that is say my 6 my third hidden layer h 3 and what that do is input to the third hidden layer is going to be z 2 and this is going to predict down z 2 hat itself and this learning algorithm will still be going down in the same ways as that, I am able to get down to the best point and my z 2 hat is perfectly coming down then again I chop off this w 3 over here the w 3 prying. So, which connects on my h 3 to z 2 dashed and then I can get down a latent space called as z 3.

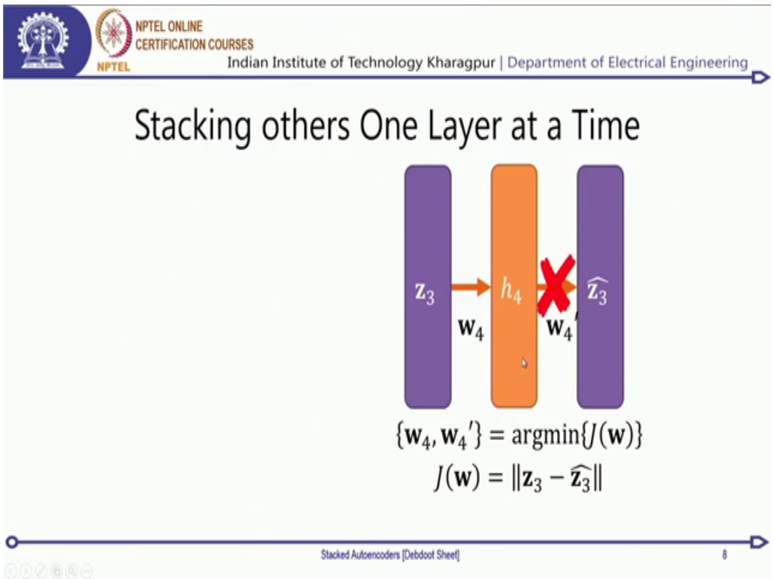
So, this by this time we have is that each single patch x which was represented in a transform domain representation called as z 1, that could be represented in a transform domain representation called as z 2 and accordingly that went for word to be represented in a transform domain representation called as z 3 and that is how I am going down.

(Refer Slide Time: 09:21)



Now, the point is I can have another layer as well and then in the similar way train it out and go it.

(Refer Slide Time: 09:28)



Now, after that when I would like to create this total multilayer perceptron in order to get down my y hat or my predicted class label or classification output whatever then what I would try to do is, I have this w 1 which was trained and kept down with me I have my w 2 which was also trained and then if you basically unroll the network in terms of all of your z 1 z 2 z 3. Then you would see that for each 4 the input was z 1 which was an output from h 3, the input to h 3 was z 2 which was an output from h 2, now input to h 2 was z 1, which one was an output of h 1 and then input to h 1 to this x.

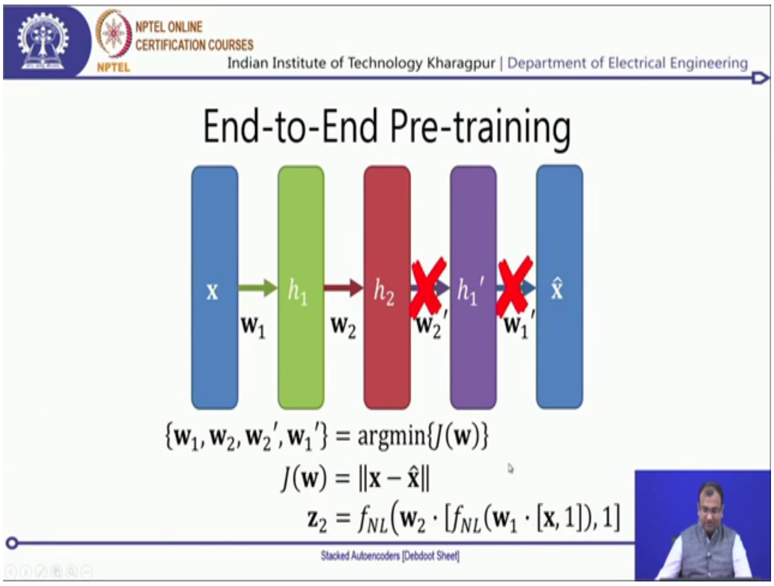
Now, if this is completely unrolled then represented in terms of a network than this is the sort of network connection which you would be getting and these weights are what will be connecting down each of these one layer to the subsequent layer; however, you see that we have been able to till now in unsupervised framework where you would not need any kind of a class label for every x that you have a y, but till now we had never been using that y and that was the beauty of using an Autoencoder for representation learning. So, w 1 w 2 w 3 and w 4 these are what are the learnt representations and they do not make use of 5.

But in order to get down y had which is my prediction I will have to connect this final output. So, z 4 or the output of the fourth hidden layer is what has to be connected down to my final decision node over here. Now that I am connecting this one I will have to initialize this w 5 and then the training process go something like this that I would try to refine all of these weights.

Now, these weights w 1 to w 4 these are trained already they have been trained. So, they are somewhere close to the global optimum is the assumption. So, I put down all the weights copy them from my earlier pieces of network and paste it here for w 5 I would be putting a random initialization over there and then start this optimization process. Then this optimization, now I am no more looking back into x, but now my idea is that the total goal to achieve is that I have the minimum error in classification and that is why I am using y as my classification ground to tensor and y dashed y hat is what is my predicted classification tensor. And now if I am able to get down a 0 error, then I am at the perfect classification using this one. So, that was one technique.

Now going down to the other technique of this one is something which is called as a end to end pre training.

(Refer Slide Time: 12:01)

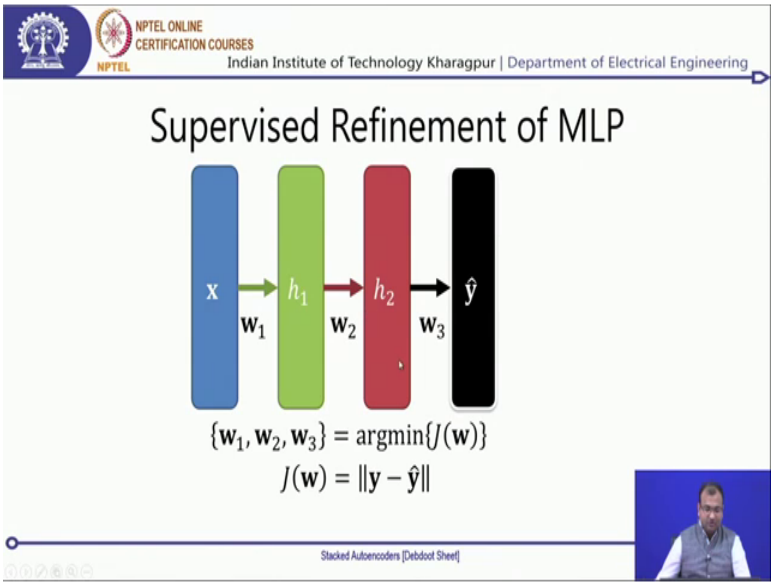


So, in end to end pre training what you would do is say that I have 2 hidden layers h 1 and h 2 and they are connected down by w 1 and w 2 and on the converse of it is what I call is from h 2 whatever is connected is called as h 1 dashed via weight of w 2 dashed, from h 1 dashed it connects and it creates x hat while these weights of w 1 dashed.

Now this will be my first training algorithm and if you look over here since it is trying to predict it is itself it is the patch itself. So, it does not make use of any supervision over here you do not have class label given down supervisor acting as to giving it is classification performance and that is a reason why this is also called as a unsupervised learning algorithm or unsupervised pre training. Now once this is done you can train and then chop of these 2 weights over there. Now once that chopped off what you are left with is this h 2 and the output from this h 2 which from the earlier cases we can relate and also called as z 2.

Now the idea is that this is the output z 2 which can be defined in something of this form.

(Refer Slide Time: 13:14).



Now, I want to do a final supervised refinement and; that means, that after the h 2 I will have to come down to a decision when coming down to a decision. So, what I need to do is I would need to preserve my w 1 and w 2s from the earlier case; w 3 will be what is reinitialized now in order to map down this h 2 on to this y. And the final classification is what will be going down through this particular form in which you try to minimize this argument.

Now, look if you compare this particular method of a n 2 and 3 training network with your earlier method which was a ladder wise pre training. One significant difference you would see in the ladder wise pre training, since we were training one weight at a time. So, the good thing is that in order to train down this first weight w 1 then w 2 you would be needing less amount of memory as such at any given point of time. So, your total ram space which is required for training is much lesser whereas, over here since you are trying to train in all the hidden layer weights.

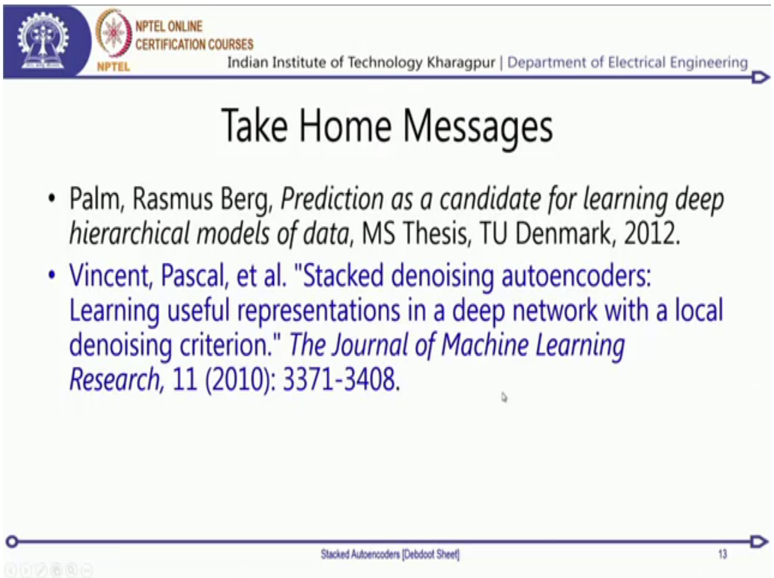
So, it is going to be much higher and also this kind of a pre training the major disadvantage is that say my final one is just with 2 hidden layer which means that I just have 3 layers of it is, but then when trying to train down the auto encoder as if you can get down into the previous slide what you can see is that I have h 1 and h 2 which connects down with w 1 and w 2, but conversely will also have to put down these 2 more weights which are w 2 dashed and w 1 dashed, now what that would impose is that if I have. So, whatever be the number of same matrix sizes for my hidden layer for my representation learning I will need twice that number over here.

So, for just 2 hidden layers where I was supposed to have just 2 set of weights for my representation learning and once at a classification, while just doing this auto encoder part I will need 2 times the number of hidden layers as the weights which is clearly large. So, if this number of hidden layers over here changes from h 1 h 2 to h 3 which means 3 hidden layers and on the converse side of it I will have h 2 dashed and h 1 dashed together.

So, that would mean that I need to have w 1 w 2 w 3 w 3 dashed w 2 dashed w 1 dashed while doing an Autoencoder training, which means that for 3 hidden layers I will need 6 sets of weights whereas, for 3 hidden layers and just doing a classification I am supposed to go down and get down only 4 set of weights.

Now typically that would mean that within an end to end learning in an autoencoding framework, you would need twice the number of weights then you would be actually needing to. So, all almost in all of twice number of weights and you would be needing to train down just an simple MLP and that is clearly a disadvantage and that is one of the reasons why for Autoencoder training ladder wise pre training is something which is preferred because you, your maximum memory is always limited by the total length of the network and the maximum variable space you would not need at any point of time variable space which is more than the total length of the network which you are speaking about.

(Refer Slide Time: 16:13)



Now given all of this we come down to an end for this lecture and that is where I have a few take home messages for you. So, auto encoders as such are quite interesting thing to explore and in order to read more about them I would definitely refer you to this master's thesis from TU Denmark by Rasmus Berg Palm. So, this is one of the most comprehensively written down text in the form of understanding hierarchical models on or auto encoding models as we called them today in terms of in deep learning.

So, he also has math lab based tool works which you can give it a try though we are not covering any of those math lab based exercises and all the models which are described over there; we would be drawing a one to one correlation for each of them and using it for our purpose, within our tutorials as well and you can relate that lot of other explanatory tutorials on codes which we had done in the previous week and something which bear down a resemblance to this thesis as well. The other one is you can go down through Vincent Pascals one on one Jmlr this is about "Stacked denoising autoencoders; and that is something which we would be doing in the next class and trying to explain you more into what is stacking of autoencoders. So, stacked autoencoders is what we have done we will be doing into denoising and sparsity within auto encoders such.

So, that brings us to the end and for this one and stay tuned and for the next class we will be doing down with sparse and denoising properties of auto encoders as well.

Thanks.