

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

Lecture - 11
Autoencoder for Representation Learning and MLP Initialization

So, good morning and we start with the next week. And so here, we would be getting started into understanding; what is a first basic structure of what is called as an Autoencoder. And this is the first family of deep neural networks which, on which we will be starting. So, it builds up on top of what we have done till last week and that was a multilayer perceptrons.

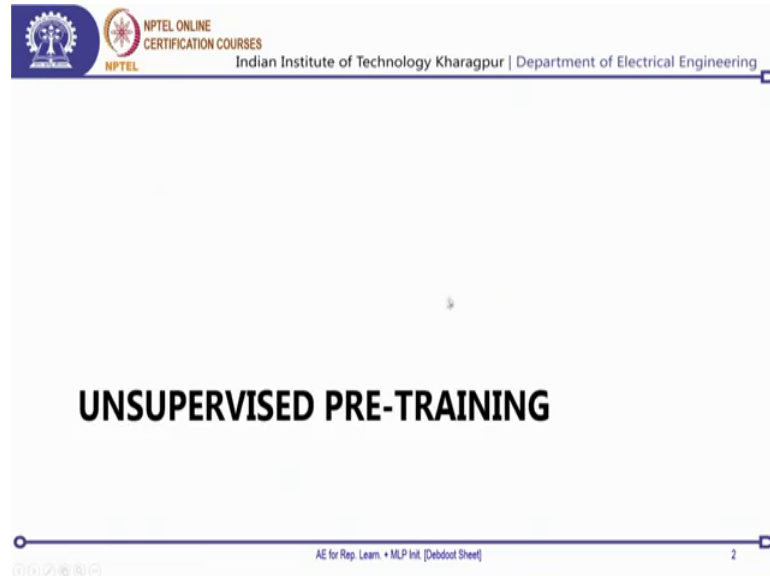
So, while typically with multilayer perceptrons, what you have learnt and was more of from a pattern space. And then, eventually there was an extension drawn down as to, if you have an image directly, then can you feed it down to solve a problem as well and that is where the deep learning comes to play. So the idea is that, as you cascade a cross the depth of a perceptron, a multilayer perceptron from one hidden layer to the next hidden layer to the next hidden layer across it is family of transformation.

So, there will be linear combinations and then an all linear transform function; together you would be ending up learning down a lot of attributes. And this is what would essentially help you in doing an image to a final classification coming down. So, what I would start down is today called as Autoencoders for Representation Learning and for Multilayer Perceptron Initialization.

So, how this is arranged is, more of like we start with what is defined as an Autoencoder and till that is that is a very simple name as it goes down; auto means itself and encoder means a particular network which learns to encode itself. And the way that this encoder learns to encode itself is something which helps in, helps us into something called as a Representation Learning. Or this is one of those family of things where you learn to extract features from the images itself. So, no more you would be needing down say heart, hand coded texture descriptors like local binary patterns or wavelet us or co-occurrence matrices, but these networks themselves will be learning.

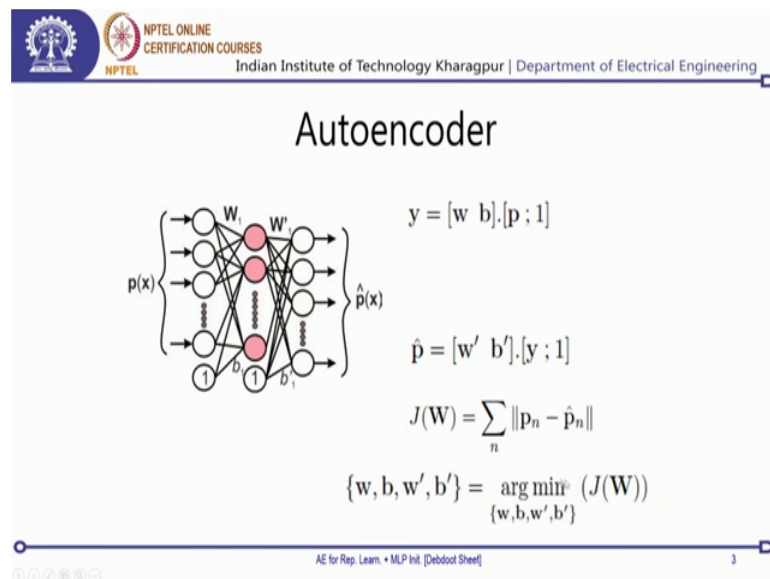
So, they will be adapting them self in order extract features. And that is what is also called as representation over there.

(Refer Slide Time: 02:16)



So, without much of a delay, let us get started. So, the first part of it is what is called as an UNSUPERVISED PRE-TRAINING. And this is again looking it down from a classification perspective in total.

(Refer Slide Time: 02:25)



So, a general structure of an Autoencoder is something which looks like this. So, what it has is, a set of neurons which will give you that input pattern to itself and then another

set of neurons which generates an output pattern. The number of nodes in the output neuron and the number of nodes in the input side over here, they are same. So, the number of neurons in the input side is equal to the number of neurons in the output side. And in essence, as you see that input over here is called as $p \times x$ for all generality and to keep it very specific to our visual computing perspectives, we will say that p of x is actually a patch; a patch at a location x .

So, if you have any kind of a coordinate location given say 300 cross 300, 300 comma 300, x equals to 300 and y equal to 300; On that image you can extract a patch of say 3 cross 3 size over there. So, that will include all the 8 neighbors around that particular pixel at 300 comma 300 location. You can have a 5 cross 5 neighborhood which will have a more number of pixels over there. You can have, 5 cross 5 will have total 25 pixels over there. So now, if I have a 5 cross 5 which has 25 pixels over there, then each pixel is what is an input to each of these neurons. And technically, in that case your number of neurons on input side has to be 25. So, similar the number of neurons on output side will also be 25 keeping in the same logic.

Now, that you have this input side which is called as $p \times x$, the output which generates from this output side over here is what is called as \hat{p} of x . And \hat{p} of x from our earlier all the classes which we have done, you have seen that typically you give an input x over here, you get an output. y and y is basically \hat{y} , which is the predicted value. So over here, we call this as the predicted value of the patch that is why it is \hat{p} of x . And for a simple Autoencoder how it would go down is that, you would have an intermediate layer with the hidden neurons over here, will just 1 layer which will have all the neurons which are called as the neurons in the first hidden layer.

And then the idea is that, given a patch in the input, it goes through all of these linear summations, then non-linear transformations. The output again comes out and goes through linear summation and non-linear transformations and this output is generated over here. And the way it is connected down is W_1 is the set of weights which connect all of these input neurons to the first hidden layer. b_1 is the set of biases which connects. W_1 dashed, so you have this dashed over here.

So, that is sort of like, the counter of W_1 which connects this hidden layers to these output neurons over here and similarly your bias also has a counter representation which

is b_1 dashed. And the way it represents is that if this was a transformation, so your output was given down as y is equal to w times b dot p_1 and your \hat{p} which comes out from here. So, y is the output of the first hidden layer over here, for the output of the next one is what comes down as \hat{p} . And then the whole idea is that, you have a cost function which is called as J_W which is equal to the L_2 norm of the input minus the output over there.

So, whatever is the patch that is a . So, here I am assuming that there is no non-linear transformation like sigmoid or over here. It is just a linear weighted summation through which it goes down. And finally, the whole training of this network can be achieved by doing some sort of a way of minimizing this cost function over here; and that is exactly what we want to do.

So in the last classes, we have studied about gradient descent which is one of those very useful methods for optimizing these weights of the network and getting it down to this particular form which is expected.

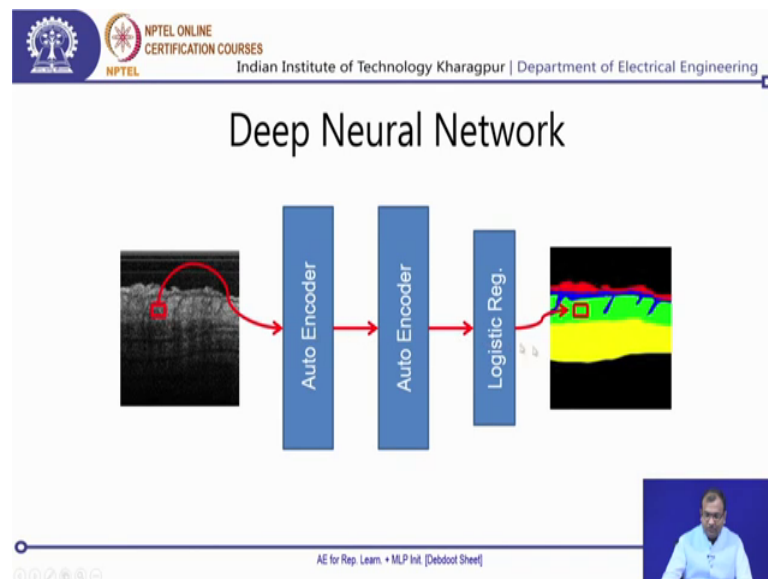
(Refer Slide Time: 06:24)



Now from there, where we go is something called as a Supervised Refinement. Because what you have studied till now is, where your inputs get represented in a way in to your intermediate layer such that your output can be reconstructed from that intermediate layer things. But then, reconstruction is not what we are looking at. So, there are multiple uses. So, some down the line we will be looking at, what happens if we are not looking at

a classification, but can we use this kind of an Autoencoder for other purposes. So, we will be coming down across with denoising autoencoders in order to find out a very practical use of them as well that, for the perspective that here we are looking at classification as problem and that is the first getting started from where we are doing. So, the next part is to go a Supervisory Refinement.

(Refer Slide Time: 07:11)



So here, typically the task is something like this, that given that I have a patch, so I take this patch over here which is red marked. And then I pass it through one Autoencoder, the second autoencoder. And next, I pass it down through a logistic regression or say a simple sigmoid decision rule over here. Now once it goes down through this decision rule, it gives me an output for a probability of a patch.

So, here how it is encoded is that, I have a, I consider a patch which is centered at a given pixel and then I do a raster scan which is I linearly translate across all the pixels over here. For any kind of a given pixel, I take a neighborhood; I feed it to my like subsequent 2 Autoencoders over here hierarchically connected. Then the output of the second Autoencoder is what is fed to the logistic regression. And then for that particular pixel, I get down a probability value and from there I can do run a classification will as to it belongs to which class.

So, it can be either the black class, the red, the blue, the green or a this yellow class; anyhow, one of these 4 classes to which would it belong. And that is the color over here

is basically which is the maximum class in the probability which comes. So, this is a result of that r max which is pasted over here. So, that is how you keep on sliding this window over the whole net, over the whole image and accordingly you will be able to get down these probability values and class labels obtained across the whole image. Now given that, this is what you would be doing, let us go into the math part over there.

(Refer Slide Time: 08:39)

The slide is titled "Weight Refinement" and is part of an NPTEL Online Certification Course. It features the following content:

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

Weight Refinement

$$t = f_{NL}([w_3 \ b_3] \cdot [f_{NL}([w_2 \ b_2] \cdot [f_{NL}([w_1 \ b_1] \cdot [(p+r) \ 1]^T) \ 1]^T) \ 1]^T)$$

$$J(W) = \sum_m \|t_m - \Omega_m\|$$

AE for Rep. Learn. + MLP Init. (Deebot Sheet)

So, what I call this output is something as a variable t . So, this t is basically a 1 hot vector. Or in simple terms, if have a 5 class classification problem, for each pixel I will get down 5 entries of which 4 entries will be 0 and one of these entries is going to be 1

The entry which is going be, so it is a 5 cross 1 matrix basically. t is a 5 cross 1 matrix for our example which we are considering. And output whichever is 1 is the particular class which is represented over here. And this t is often called as the target tensor or for classification purposes. So, here what I do is, you remember that we had 2 Autoencoders connected, so this $w_1 \ b_1$ is for the first Autoencoder. The output of that one is what is generated with this f_{NL} or the non-linear transformation.

Now, that goes as input to the second Autoencoder that is why you have this feed over here or dot product with the second order encoder weights w_2 and b_2 . The input to the first autoencoder is what is this patch p and we also put down some a random number called as r . Now the significance of this random number is immense, but it is not within the scope of this first lecture where we would be studying this random number over here.

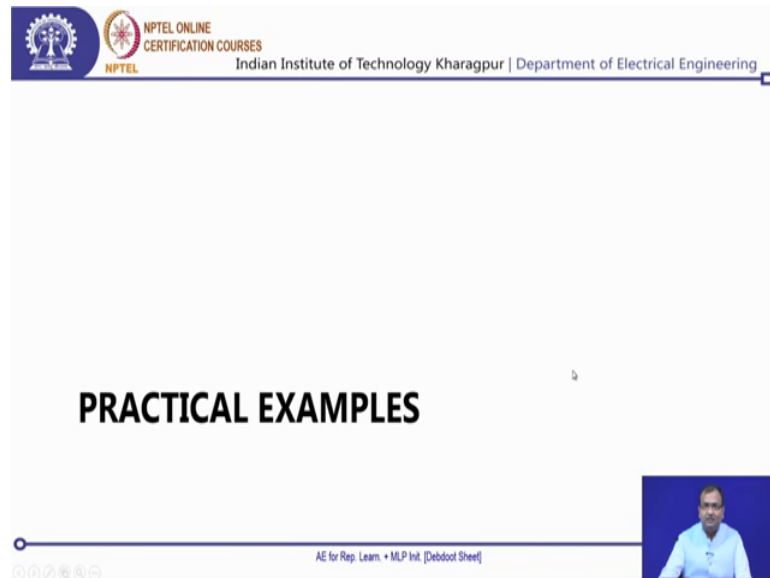
So, as of now you can even assume this random number to be 0. So, for our lab tutorial examples, we will be taking this random number as 0 and then we would be proceeding and eventually you would see how a particular distributed random number does come into a important play. And that will also make us realize a very important aspect about machine learning is that randomness is a very important factor. And in fact, that is something which will help you avoid in a lot of cases getting locked into local minimas. So, here we take the same thing.

So, now that it keeps on going in this final layer the w_3 is what is connect the second autoencoder to the logistics regression. And this is what has to be trained. Now what we would do initially is that, initially we train for w_1 and w_2 in an autoencoder mode. Now once this w_1 and w_2 is trained in autoencoder mode so, in that kind of a training process, you never the class labels coming down and these are what are the feature extraction layers which are trained perfectly. Now when that feeds on to this w_3 this is what has to trained into a classification mode, w_3 and b_3 . For here, when we do this weight refinement concept over here, what we do is w_1 and w_2 are already trained, they will not be changing drastically, but there would be minor shift.

So, may be 1 percent or 2 percent of the values in w_1 and w_2 would be changing. A majority of these values in w_3 which were randomly initialized at the start of it, will definitely be changing. And together is what you get down this output. And then, the final cost function over here is what is defined accordingly.

So, this is, ω_m is basically my ground truth and t_m is what I am getting down over here and m is basically the m th patch or the m th sample for which I am making a decision over here. So, while t is a 1 hot vector, ω is also a 1 hot vector and this is what is used for my Weight Refinement.

(Refer Slide Time: 11:46)



The slide features the NPTEL logo and text at the top: "NPTEL ONLINE CERTIFICATION COURSES" and "Indian Institute of Technology Kharagpur | Department of Electrical Engineering". The main content is the title "PRACTICAL EXAMPLES" in large, bold, black letters. At the bottom right, there is a small video inset of a man in a light blue shirt. The footer contains the text "AE for Rep. Learn. + MLP Int. [Debdoot Sheet]" and navigation icons.

Now, let us get into some of these PRACTICAL EXAMPLES. And again given the fact that I come down from the field of medical image analyzers, lot of these examples I would be taking down including the one which I had shown down are from medicines. So, the earlier one is what is called as optical coherence tomography and we would be doing down more within.

(Refer Slide Time: 12:04)



The slide features the NPTEL logo and text at the top: "NPTEL ONLINE CERTIFICATION COURSES" and "Indian Institute of Technology Kharagpur | Department of Electrical Engineering". The main content is a citation: "Vincent, Pascal, et al. 'Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.' *The Journal of Machine Learning Research* 11 (2010): 3371-3408." Below the citation is the title "HANDWRITTEN DIGIT RECOGNITION" in large, bold, black letters. At the bottom right, there is a small video inset of a man in a light blue shirt. The footer contains the text "AE for Rep. Learn. + MLP Int. [Debdoot Sheet]" and navigation icons.

But, let us start with the very simple one, which is from a classical paper on HANDWRITTEN DIGIT RECOGNITION. And this is by Vincent Pascal. And so this

was in the journal of machine learning research in 2010. And that is called as Stacked denoising autoencoders, and how do learn useful representations in a deep network with local denoising criterion.

So, while we will be discussing bit more on this local denoising criteria at a later on stage, but before that the most important part is that if we have an autoencoder, then can we use it for even digit recognition kind of problem.

(Refer Slide Time: 12:42)



The slide 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 'About the Challenge'. To the left is a 10x10 grid of handwritten digits. To the right is a bulleted list of challenge details. At the bottom right is a small video inset of a speaker.

1	5	8	6	5	7	3	8	6	1
3	2	5	8	2	4	4	1	5	
6	5	4	4	3	1	8	3	8	6
6	1	0	0	4	3	7	4	7	5
7	1	9	0	8	8	3	3	2	9
5	4	9	2	6	8	1	4	0	5
7	3	8	6	1	3	3	3	5	3
3	3	6	1	7	2	5	7	6	5
0	3	1	2	3	0	6	5	9	3
4	3	7	6	9	4	3	1	3	5

- MNIST
 - Handwritten digit recognition challenge
- Train set
 - 60,000 samples
- Test set
 - 10,000 samples
- Patch size
 - 28 x 28 pixels
 - uint8 gray valued
 - Binary images

AE for Rep. Learn. • MLP Int. (Deebot Sheet)

So, what this challenge is, this is one of the practical problems we will be solving in the lab class in the subsequent lectures as well. So, the idea is that, you have this handwritten digits over here. So, you see this is a one this is 5, this is 8, 6, 5, 7, 3, 8 and this looks like a 6 and this is a 1. So, this also looks like a 6, but again it is a bit occluded over here. Now the objective of this challenge is that, say you have these small patches, so this is one patch or one single image given down and this is what is present within the patch. Now you have to classify what is written down.

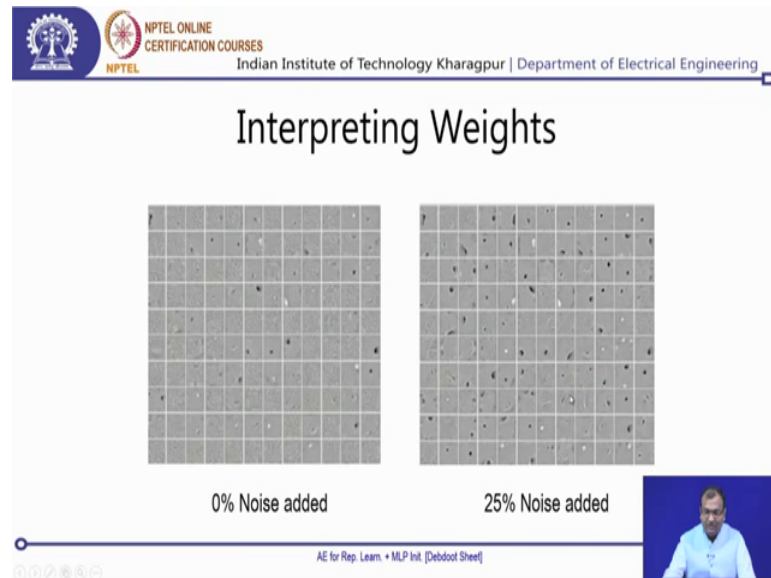
So, I have basically 10 digits which are 0 to 9. So over here, my target classification is going to be a 10 class classification. And accordingly, my t vector or my target vector that is going to be a 1 hot vector of size 10 cross 1 right; I mean this is pretty intuitive from here. Now what we need to do is, we need to understand a bit more about this one. So, this is what is called as the MNIST data and it is from the handwritten digit.

So, MNIST is from the modified version of the NIST or the National Institute of Standards And Technology. So, this is from an agency based in the US. And they provide out this data set for doing these standardized classification tests. So, the training set consists of 60000 samples. And they are distributed equally across all the 10 types of digits. The test set over there consists of 10000 samples. So, each class of digit has 10, has 1000 examples basically over there. So, the idea is that the samples which you see on the test set are not present in the training set and vice versa and that would make it independent of each other.

Each of these image patches which you have, where you have this digit written down, that is of a size 28 cross 28 pixels. Each is a 8 bit gray valued pixel container. So, they are in the range of 0 to 255, but they are all binary images; it means that if it is white then that pixel is 255.

Where ever it is gray that pixel is 0 that is how it is represented. So, it is technically a binary image, but they do not store it into a binary form for some of their storage reasons is in order to make it compatible and easily viewable across most viewer which do not support a binary, but would support an 8 bit value. So, they store it just in a 8 bit form. So, we have download links, but it is easy to Google down and come down. So in the lab class, we will be showing down how to get started with. And these are very standard for most of the getting started examples and you would see them as the first few examples which come up.

(Refer Slide Time: 15:26)



So, once you train down that network and everything, what comes down is a pretty interesting part over there.

So, one was like, if you look into this weights of the first layer, then you need to make a sense of what is happening over there. Now, since this paper was on basically denoising, sorry denoising autoencoders and they added down local noise as local perturbations over there to see what are the effects. So, what happened was that with addition of noise it was ending up learning better features.

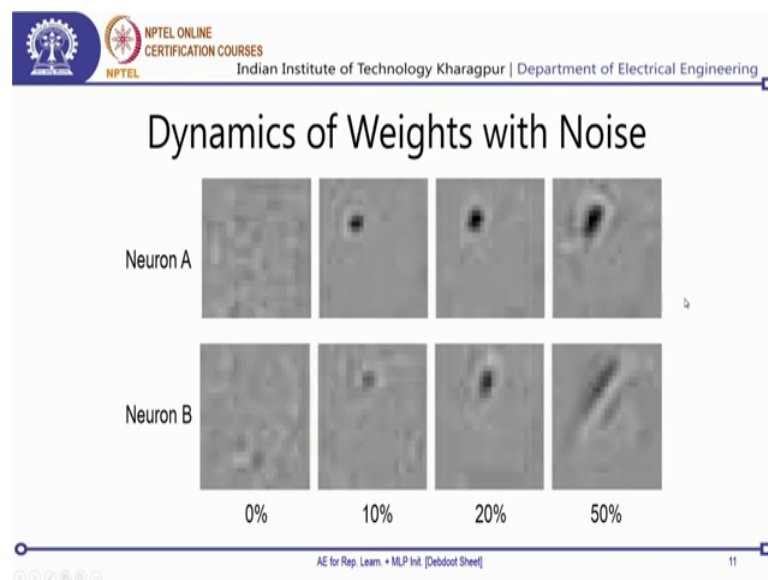
Now these patches which you see over here, these are patches these are basically rectangular form of representing the weights which connect down the input layer to the first hidden layer and that is the for the first autoencoder. So, if you, so basically you have a 28 cross 28 pixels over there. So that means that from there if I connect down to my say 4, I remembered something like 400 neurons are present on the first hidden layer. So, that would mean that, there are basically 400 weights which connect down each neuron to the first neuron of the hidden layer. Then there are 400 weights which will connect down all inputs to the second neuron.

So, this is the first neuron, this is the second neuron and this is arranged as a collection of 20 cross 20 weights over there which will beat 400 such links which are connected. And if you count over here, there are basically 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 so, this is just 12 cross 12 representation which they have given. But, since this is 12 cross 12. So, this

would mean that my input side of, my hidden layer over there will be 144 in that case. And so, inside each of this is what is sized as the same as that patch over there. So, the patch size was 28 cross 28. So, this is a 28 cross 28 patch which I have. And then, these are the weights which connect down.

So, this is a link between 1 input pixel to a given a layer in the hidden neuron. Now these we will do a bit of more complicated mathematics in the subsequent classes where we would learn down relations between these appearance weights to a correlation function itself, but as of now what you see is, that they represent some sort of wavelet us. So, these mimic down your Gabor wavelet us to a certain extent and this is what where it comes down in terms of learning it.

(Refer Slide Time: 17:49)



So, there are more interesting facts over here which are like, how do they learn different aspect with different degree of noise. So if you see, initially if there is no noise added down, it get appears not to learn down anything whereas, if you have 50 percent noise added down then you would see that what 2 different neurons are learning are pretty different in appearance from each other, while this looks like some sort of a Gabor wavelet which is shifted and up, this looks like a Laplacian wavelet which comes down over here. So, these are interesting facts that it can learn down from data itself.

(Refer Slide Time: 18:20)

The slide is titled "Appearance Model Variations" and is part of an NPTEL ONLINE CERTIFICATION COURSE from the Indian Institute of Technology Kharagpur, Department of Electrical Engineering. It displays a grid of images illustrating different variations of handwritten digits. The grid is organized into two main sections: (a) and (b). Section (a) shows digits with random backgrounds and rotations. Section (b) shows digits with rectangular and convex hull occlusions. A small video inset of a speaker is visible in the bottom right corner.

(a) rot, bg-rand, bg-img, bg-img-rot



(b) rect, rect-img, convex

AE for Rep. Learn • MLP Int. [Debdoot Sheel]

Now from there, we can get into another interesting problem which they have shown about appearance model variations. So, the idea was that, we had images where the number was written upright and given, but then if you apply some degree of twist and turn around with the number, then can the machine still learn to recognize or say that the background is not black at all, but the background is noisy itself.

Then if the background is textured, then if we have some sort of a rotation on a textured background this is where you have a convex hull imposed on top of the number and given down, this is where you have a patch occluding, this convex hull appears as a patch and that occludes on top of something.

(Refer Slide Time: 19:00)



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

Performance Metrics

Data Set	SVM _{rbf}	DBN-1	SAE-3	DBN-3	SDAE-3 (v)
<i>MNIST</i>	1.40±0.23	1.21±0.21	1.40±0.23	1.24±0.22	1.28±0.22 (25%)
<i>basic</i>	3.03±0.15	3.94±0.17	3.46±0.16	3.11±0.15	2.84±0.15 (10%)
<i>rot</i>	11.11±0.28	14.69±0.31	10.30±0.27	10.30±0.27	9.53±0.26 (25%)
<i>bg-rand</i>	14.58±0.31	9.80±0.26	11.28±0.28	6.73±0.22	10.30±0.27 (40%)
<i>bg-img</i>	22.61±0.37	16.15±0.32	23.00±0.37	16.31±0.32	16.68±0.33 (25%)
<i>bg-img-rot</i>	55.18±0.44	52.21±0.44	51.93±0.44	47.39±0.44	43.76±0.43 (25%)
<i>rect</i>	2.15±0.13	4.71±0.19	2.41±0.13	2.60±0.14	1.99±0.12 (10%)
<i>rect-img</i>	24.04±0.37	23.69±0.37	24.05±0.37	22.50±0.37	21.59±0.36 (25%)
<i>convex</i>	19.13±0.34	19.92±0.35	18.41±0.34	18.63±0.34	19.06±0.34 (10%)
<i>tzanetakis</i>	14.41±2.18	18.07±1.31	16.15±1.95	18.38±1.64	16.02±1.04(0.05)

AE for Rep. Learn. • MLP Init. [Deebot Sheet]

13

And then together using all of these examples, this is what they have found out. So, SDAE is the version which they were working down. And these are the Performance Metrics in terms of errors.

So, if you look in terms of errors, you would see that with the basic MNIST, they have one of the lowest a very low error, but not necessarily the lowest error. The good thing is that, as you keep on increasing all of these weird transformations on top of it, you would see that the lowest error is something which you would be getting down consistently with this denoising autoencoder. And that is something which brings us to the fact that, these autoencoders are basically a very good family of learning down representations and that is how they are able to overcome most of this challenges.

(Refer Slide Time: 19:43)

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

Shin, Hoo-Chang, et al. "Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data." *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35.8 (2013): 1930-1943.

ORGAN DETECTION IN 4D MRI

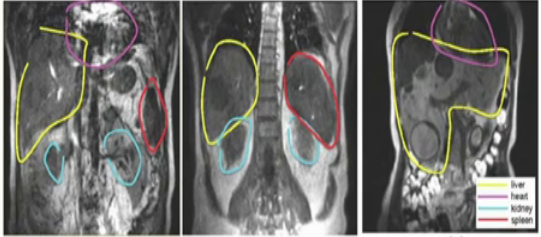
AE for Rep. Learn. • MLP Init. [Deboot Sheet] 14

The next, I would be taking down is on ORGAN DETECTION IN 4D MRI.

(Refer Slide Time: 19:49)

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

About the Challenge

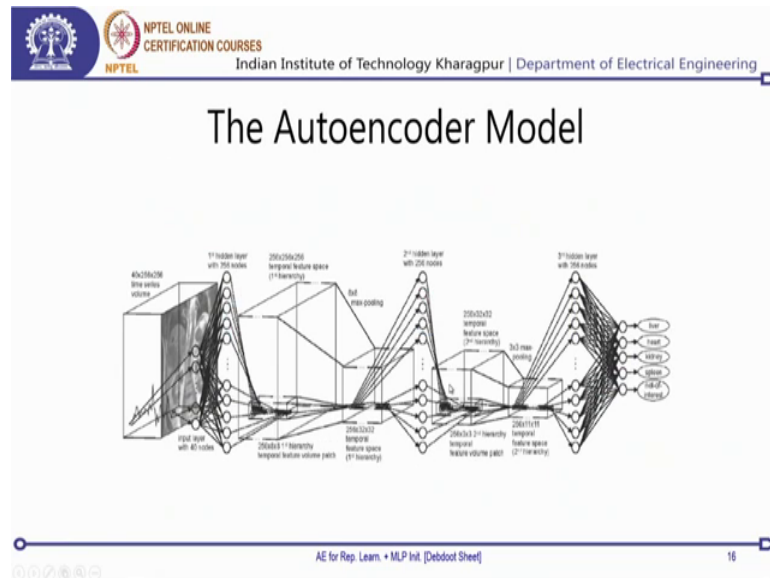


Dataset A: 256 x 256 images, 7-12 contiguous coronal slices, 40 temporal points, Patients # 46
Dataset B: 256 x 247 images, 14 contiguous coronal slices, 40 temporal points, Patients # 3
Dataset C: 209 x 256 images, 12 contiguous coronal slices, 40 temporal points, Patients # 29

AE for Rep. Learn. • MLP Init. [Deboot Sheet] 15

So, we will be doing this kind of an exercise a bit later on, but this is just a fancy example. So, here the idea was that, you had 3 different data sets on which the point was to identify pixels corresponding to the tissue. But what was marked and given was these kind of bounding polygons, free hand bounding polygons which represent each different kind of a tissue.

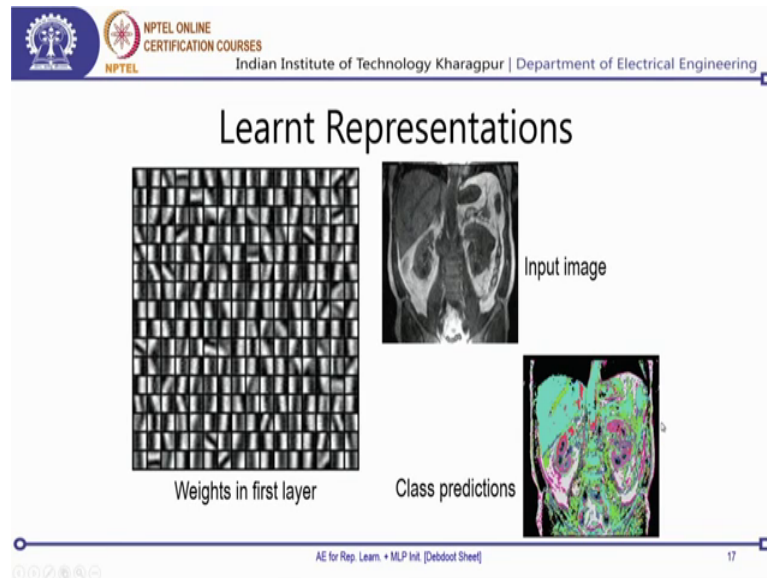
(Refer Slide Time: 20:11)



Now, what the authors had done is that, they had constructed a very densely connected autoencoder and this autoencoder was what was operating on the volume space. So, this was no more 2D pixel space. So, not a patch which consists of a say 28 cross 28 sized over there. But since it is in 3D, because there is a x dimension, y dimension and z dimension over there. So, you end to take a volume over there, a sub volume on which it is connected.

So, the number of neurons over here is still larger. So, given the fact that, if you have a 5 cross 5, you have 25 neurons which exist in normal 2D space. In 3D space, you would make an if few vocal translations to a 5 cross 5 cross 5 and that would make it 125 neurons. Now that is bigger of 1 order. So, you increase 1 number it order, it increases in the cube of that number basically.

(Refer Slide Time: 21:02)



Now, using that what ended up was that, they try to visualize this first layers and what came out was that these look something like edge detectors basically.

If you remember your sobel kernels, then these appear like sobel kernels, but a rotated version, then shifted versions of them. And accordingly, using just one single layer of representation learning, you could a very decent, an accurate prediction of the different tissue classes present in an MR image itself. So, that is the power of deep learning and where it goes down that, you just need to make a network, find a way of packing the data and training it out efficiently. And the rest of the features and what to do and how to associate a classification rule with a particular kind of a feature is what gets done into went within the system itself.

(Refer Slide Time: 21:52)

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

Sheet, Debdoot, et al. "Deep learning of tissue specific speckle representations in optical coherence tomography and deeper exploration for in situ histology." *IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, 2015.

OCT TISSUE CHARACTERIZATION

AE for Rep. Learn. • MLP Init. [Debdoot Sheet] 18

So, this is my personal work from 2015 and we will be doing this as one of the exercises as well, is on tissue characterization.

(Refer Slide Time: 21:57)

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

About the Challenge

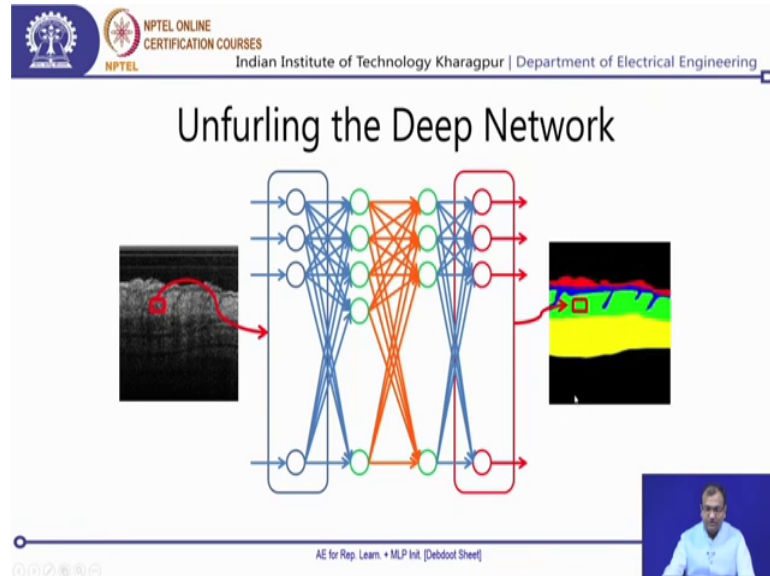
- **Data Collection**
 - School of Medical Science and Technology, Indian Institute of Technology Kharagpur
 - 1300 nm (HPBW 100 nm) Swept Source OCT System
 - OCS 1300 SS, ThorLabs, NJ, USA
 - 8 bit bitmap images
 - Histology for ground truth
 - HE stained
- **Samples**
 - *Mus musculus* (small mice)
 - 16 healthy skin
 - 2 wounds on skin
- **DNN architecture**
 - Patch size – 36 × 36 px
 - DAE1 – 400 nodes
 - DAE2 – 100 nodes
 - Target – Logistic Reg.
 - 5 outputs
 - Sparsity – 20%
 - Mini-batch training
- **In situ Histology Performance**
 - Epithelium – 96%
 - Papillary dermis – 93%
 - Dermis – 99%
 - Adipose tissue – 98%

AE for Rep. Learn. • MLP Init. [Debdoot Sheet] 19

So, the exact network which I was showing is over here. So, what you have is, we take down patches of 36 cross 36 pixels and each of these are gray valued patches. You have 2 autoencoders, one of them with 400 neurons, the next one with 100 neurons and then you have a target with the 5 outputs. So, there is a part of sparsity, we will be learning that

subsequently as well. And including all of this, you get down an performance accuracy which looks something like this which is typically a very high accuracy as such.

(Refer Slide Time: 22:29)



So, the idea was that, you have a patch coming down and you feed it down to your input neurons, just linearize. So, you had 36 cross 36, you make 36 square number of neurons over here. You have your bias added down, then you have the first hidden layer which had 400 neurons, then the second hidden layer which has 100 neurons and from there you have 5 neurons on the output side of it. And then based on whichever is coming as a 1 hot, output this color is associated over there.

(Refer Slide Time: 22:55)

The slide features the NPTEL logo and text at the top: 'NPTEL ONLINE CERTIFICATION COURSES Indian Institute of Technology Kharagpur | Department of Electrical Engineering'. The main title is 'Learning of Representations'. On the left, a diagram shows a neural network with three layers of nodes (input, hidden, and output) connected by lines. On the right, a large grid of small grayscale images shows speckle patterns. Below the grid, the text reads 'Representation of speckle appearance models learned by DAE1'. At the bottom right, there is a small video inset of a person. The footer contains the text 'AE for Rep. Learn. • MLP Init. [Debook Sheet]' and navigation icons.

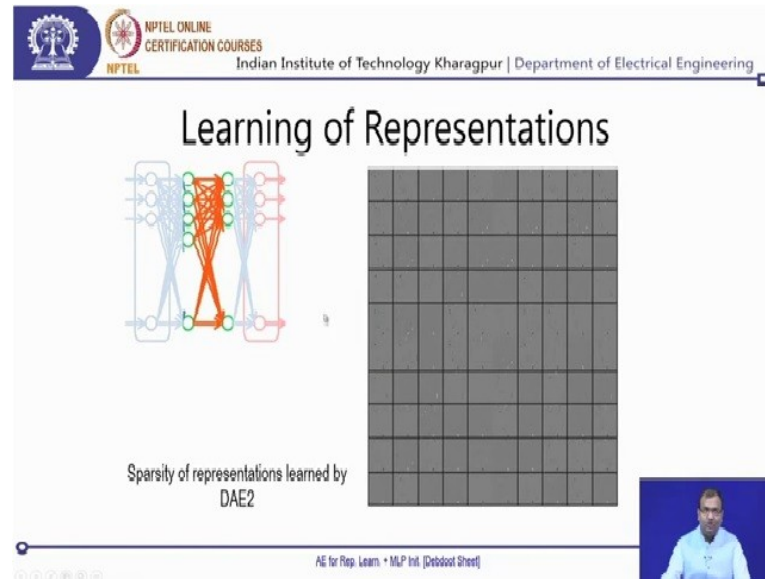
Now, we wanted to actually look into what the network was ending up learning. So, this is the interesting fact over there. So, if you arrange these weights of the first, the connection between the new, these input neurons to the first hidden layer, then this is what the arrangement looks like.

Now, each of this is 1, 1 of these. So, say for this first neuron over here, this is all the connections from all the input neurons to the first neuron of the hidden layer, this is for the second neuron of the hidden layer, this is for the third neuron of the hidden layer. Now if you look at these patches, since these are the some sort of weights and how what they have learned.

Now you would see that, a lot of these weights look very similar. In fact, majority of them are pretty similar and this particular weight over here is an offbeat because this is sort of an inverted version of the weight over here or the inverted version of majority of the weights. Some of these weights like these do not learn anything.

Some of these learn to do a DC shift kind of stuff; some of these learn to do some weird kind of an edge detection. Now, together with this one, what is interesting is that most of these weights being very similar. This sort of represents a very redundant system. Now given that we should be looking at the Weights which connect down the first hidden layer to the second hidden layer itself.

(Refer Slide Time: 24:03)



And that is where comes a more interesting fact so the, you see this is basically one of these patches. So, there are it is a 100 neurons. So, you have basically 10 cross 10 array.

So, this boundary somehow how is not appearing quiet perfect over here. And then each of this is a 20 cross 20 point because you have 200 neurons connecting it over here. Now you would see that most of these regions which are gray are basically 0 valued and the ones which are white are very high value, high positive valued. The ones which are black spots are high negative value.

Now what comes down is, since you did see in the earlier one that a lot of these weights were very similar so that means that, you can use any one of those representations and you do not need to use all of them. So, and that was one of the reasons why most of these weights over here came down as 0. And this is what is typically called as a very sparsely represented system because most of the values in the width matrix is 0 and very few of them are non 0 values.

We will get into more details in the subsequent ones and these are another interesting fact within Deep Neural Networks. You need to understand is that these networks are capable of learning the ample number of representations which are needed and if it will never be over fitting on to a representation in some way.

(Refer Slide Time: 25:21)

The slide features the NPTEL logo and text at the top: "NPTEL ONLINE CERTIFICATION COURSES" and "Indian Institute of Technology Kharagpur | Department of Electrical Engineering". The main text reads: "Maji, Debapriya; Santara, Anirban, et al. 'Deep Neural Network and Random Forest Hybrid Architecture for Learning to Detect Retinal Vessels in Fundus Images.' 34th Annual Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015." Below this is the title "RETINAL VESSEL SEGMENTATION" in large, bold, black letters. At the bottom right, there is a small video inset of a man in a light blue shirt. The footer contains the text "AE for Rep. Learn. • MLP Init. [Deboot Sheet]" and navigation icons.

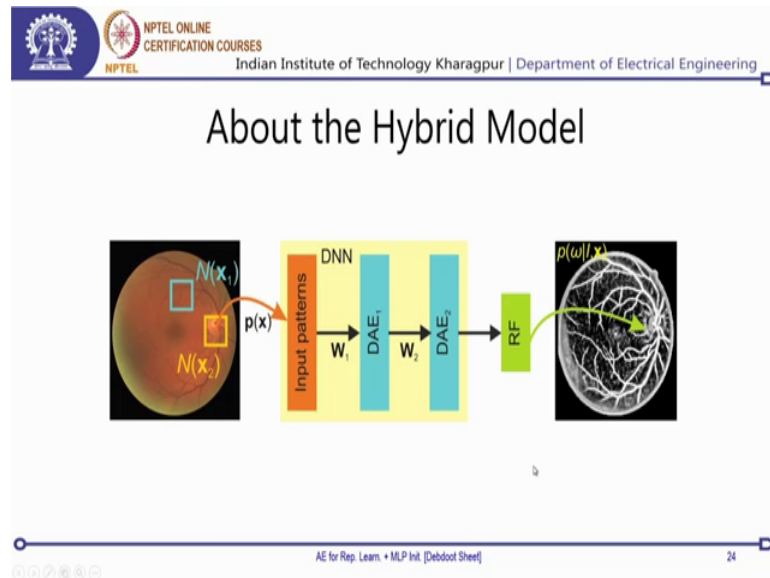
Next is another of earlier paper which is on RETINAL VESSEL SEGMENTATION, is in 2015.

(Refer Slide Time: 25:28)

The slide features the NPTEL logo and text at the top: "NPTEL ONLINE CERTIFICATION COURSES" and "Indian Institute of Technology Kharagpur | Department of Electrical Engineering". The title "About the Challenge" is centered. On the left, there are three images: a color fundus image, a binary vessel map, and a grayscale fundus image. On the right, there is a bulleted list: "• Images" with sub-points "– Color Fundus" and "– 563 x 582 pixels"; "• Train set" with sub-point "– 20 images"; and "• Test set" with sub-point "– 20 images". At the bottom right, there is a small video inset of a man in a light blue shirt. The footer contains the text "AE for Rep. Learn. • MLP Init. [Deboot Sheet]" and navigation icons.

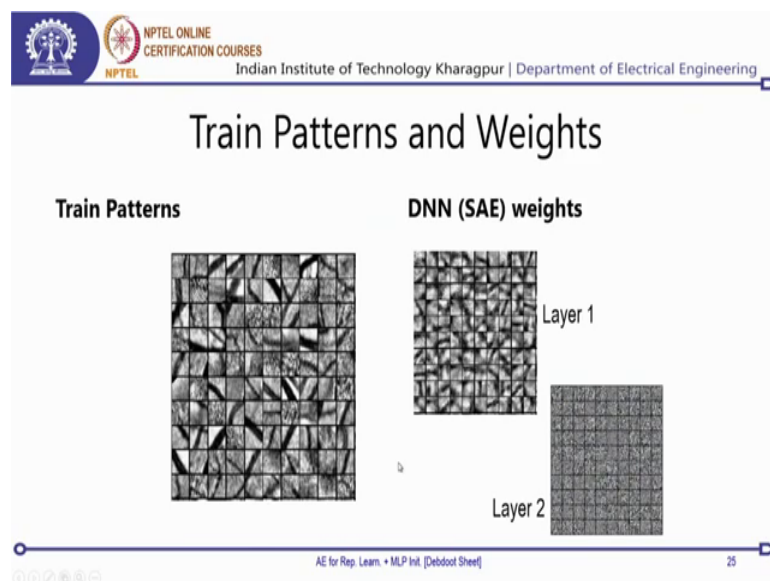
So, the idea was to get down these kind of vascular maps and this is what we came down. So, this was again, take down fundus image and you would take a small patch around over here put it down into an autoencoder and train it.

(Refer Slide Time: 25:40)



So, this is what we had done. But the idea was that, you do not put a logistics regression or a final decision over there, but you use these autoencoders for just representation learnings such that you can cascade them to a random forest and that can be used for a decision making purpose. So, this is another way of doing this in a hybrid model, where your final classifier fire may not necessarily be a neuron. So, you can use this deep neural network for just feature extraction and then subsequently padded down with a standard classifier in order to do full classification.

(Refer Slide Time: 26:14)



So, here what we did was that, we extracted down few bunch of patches and then we use these for training. So, these are the sort of patches which got extracted. And the first layer weights are what visualized and came down. And these were something which were getting tuned on to these patches itself if you look. So, these are sort of like a Gabor wavelet, but a very first order. So, a very wide ordered Gabor wavelet that your special span is very low.

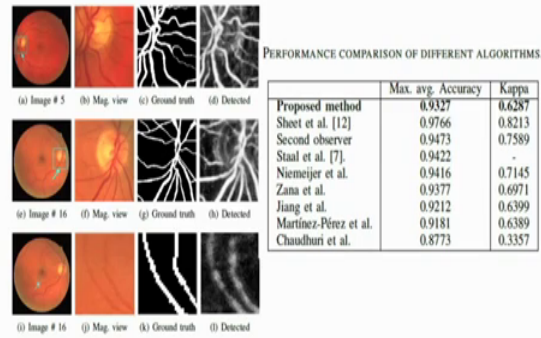
Now if you look at the second layer, interestingly over here this is no more a sparse system; this is like a very densely mapped out system because these were very directional filters which it got learnt out and that were based on the diversity of the data. So, it is not necessary that a very dirty or a very noisy looking image as in those ocids, they would be always hitting down to a very densely connected neuron.

Because you did realize that although it was appearing very noisy, but most of the features when you compress it down, they are just very few unique features whereas a very clean looking image like this retinal images over here, they rather exhibit larger amount of diversity in terms of it is representation space and that is what in influences this to be a non sparse system during classification as well. So, these are few interesting observations which we do. And as we go down through subsequent ones we will come to know more about. So, we will be discussing more in details about how to unwrap and unroll a neural network, unbox the whole thing and see what it has learned and how it is performing.

(Refer Slide Time: 27:41)

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

Performances



PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS.

	Max. avg. Accuracy	Kappa
Proposed method	0.9327	0.6287
Sheet et al. [12]	0.9766	0.8213
Second observer	0.9473	0.7589
Staal et al. [7]	0.9422	-
Niemeijer et al.	0.9416	0.7145
Zana et al.	0.9377	0.6971
Jiang et al.	0.9212	0.6399
Martínez-Pérez et al.	0.9181	0.6389
Chaudhuri et al.	0.8773	0.3357

AE for Rep. Learn. • MLP Init. [Deeboot Sheet] 28


So, together this was a final performance which comes out. And if you see that, these very thin vessels which are like rarely visible even on the image to a non-expert, these are what got tracked down very accurately in the final classification. And so these are interesting facts about representation learning that they can learn very complex patterns over a very diverse region, given that you have ample amount training data provided to this one.

(Refer Slide Time: 28:12)

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

Take Home Messages

- LeCunn, Yan, et al. "Deep Learning." *Nature*, 521 (2015): 436-444.
- Bengio, Yoshua, Aaron Courville, and Pierre Vincent. "Representation learning: A review and new perspectives." *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35.8 (2013): 1798-1828.
- Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *The Journal of Machine Learning Research*, 11 (2010): 3371-3408.



AE for Rep. Learn. • MLP Init. [Deeboot Sheet]

So finally, coming to the end is the that what you can do is you can read down through this interesting paper on deep learning which appeared in nature. It is a consolidated summary, very good magazine like reading article over there. For understanding more about representation learning and we will be doing a few exercises from based on this particular paper as well which is, you can look into this particular one on representation learning by Bengio, Courville and Vincent and transaction on pattern analysis and machine intelligence in 2013 from the special issue.

So, that makes us come to an end of this one. And in the subsequent classes, we would actually be doing more of tutorial based understanding of how to code it down and then unrolling the model to understand the math of how it is going inside. So, with that stay tuned.

Thanks.