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

Lecture – 27
Simple CNN Model: LeNet

Welcome back. So, today we would be starting back on Convolutional Neural Networks. So, in the last class I had introduced you to some of the very basic operators on convolutions. And the some of the basic structural building blocks for how a CNN works. Today we would be building up on top of what we had done in the last class and get you introduced into one of the early versions of convolutional neural network, which was referred to as a LeNet it is still it referred to as a LeNet and attributed to Yann Lecun who had invented this particular 1 and we have already solved a few examples in our autoencoder tutorials, using the animist digits classification problem.

So, LeNet was technically a thing which was invented more off from the MNIST perspective and, it was for the standard NIST database which had the challenge of identifying 26 alphabets in the English alphabet corpus in the upper case lower case, making it 52 such things to be identified and on top of it 10 different numbers and that together makes it 62 classification problem which was being solved out, then that was not a [laughter] small problem as such, but the shorter version of that and the simpler and sweeter version is what is done with the digits identification. Since each digit is quite uniquely available and has a major implication in a very practical sphere of life and that is on postal code identification.

So, without much of delay let us get started into what we are doing. So, today we would be working on convolutional neural network and, an taking down this example of LeNet over there, I would be revising a few of basic concepts which we had covered down and also coming down to a bit of numerical solution.

(Refer Slide Time: 02:03)

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

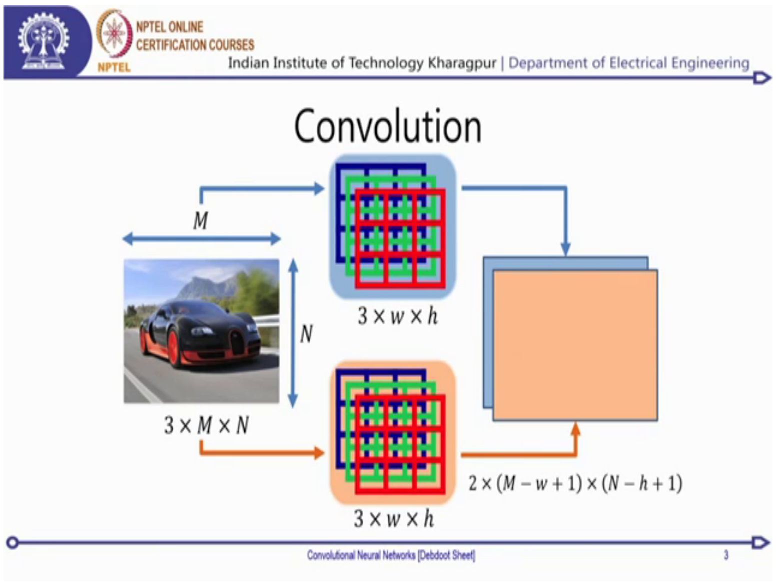
Organization

- Revising Building Blocks
  - Convolution
  - Stride
  - Padding
  - Pooling
  - Deconvolution
  - ReLU Transfer Function
- LeNet for Handwritten Digits Classification

Convolutional Neural Networks [Debdoot Sheet]

So, that it makes it easier for you to understand and grasp the concepts as well. So, this is organized as we will just be revising the building blocks, some of these building blocks which are important in the perspective of understanding LeNet and, then I would be showing them, what is the standard LeNet architecture, for handwritten digit classification.

(Refer Slide Time: 02:28)


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

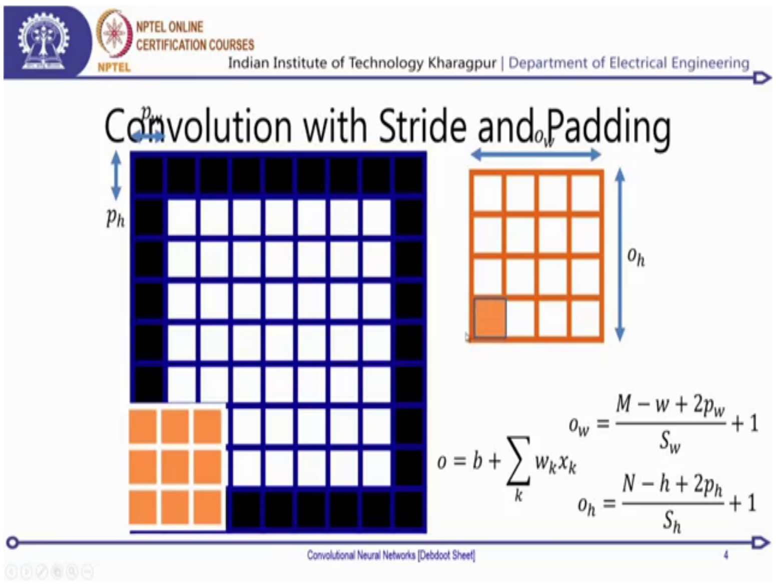
Convolution

Convolutional Neural Networks [Debdoot Sheet] 3

So, let us get into it. So, if you remember convolution how it went down was that if you have an image, which has a width of M and a height of N and 3 channels, then you have a 3 cross M cross and, if this is a grayscale image then it becomes 1 cross M cross N ok. Now the point was that we can define 1 kernel, which will have 3 different planes or 3 different channels of the kernel as well. So, this can be a 3 cross 3 kernel or a small M cross small N kernel and, and the number of channels over here since we are doing 2 deconvolution so, that will match down to the number of channels in the image space as well.

So, this becomes 1 kernel, now as I convolve with this kernel of 3 cross w cross h, then I am going to get down 1 matrix over here. Now similarly I can have another kernel and then as I convolve with this 1 I get down another matrix coming down over here. And this output matrix over here will have dimensionally t equal to so, the first dimension which is by number of channels which comes down by default definition that is equal to 2, or the number of kernels present over there, and the width over there is M minus w plus 1 and height or the number of rows is N minus h plus 1. So, this goes down pretty much by your standard definition of convolution.

(Refer Slide Time: 03:46)


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

Convolution with Stride and Padding

p_w
p_h
o_w
o_h
Convolutional Neural Networks [Debdoot Sheet]
4

$$o = b + \sum_k w_k x_k$$

$$o_w = \frac{M - w + 2p_w}{S_w} + 1$$

$$o_h = \frac{N - h + 2p_h}{S_h} + 1$$

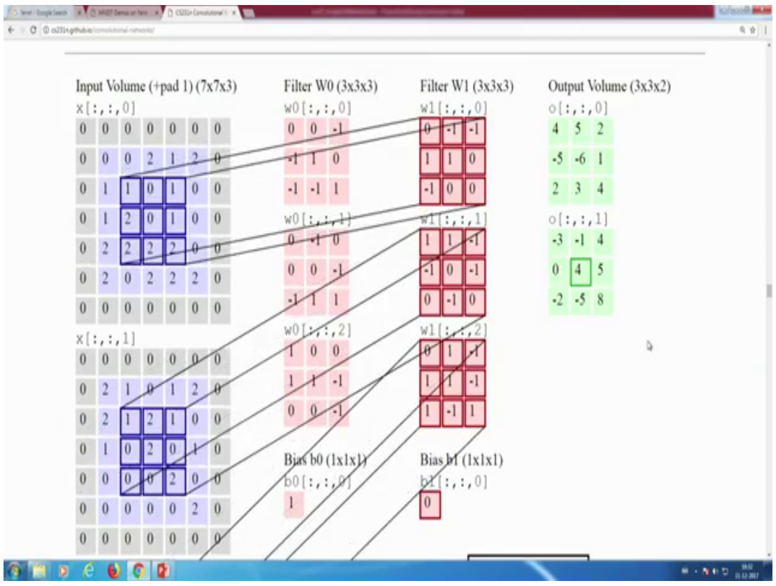
So, now when we were trying to do a convolution with the stride and padding which is a very practical case of working it out, the idea was that if you have some image over there and your, if you are trying to do down your convolution by just moving it out, then the major problem which comes down is at the borders you will not be able to operate.

So, the size keeps on decreasing from the periphery of the image itself, if we even if you are taking it with a stride of 1 in order to get rid of that what we had done was that our idea was let us padded down with 0s, which is equal to this extra number of points which just get outside. in case we want to have a full sized convolution result also coming down. So, if I have a pad height of ph and a pad width of p w and, then we start convolving this one with a stride of S and appropriately it keeps on merging and then come down.

So, as it happens together. So, this is how a resultant matrix is formed over there and, then your output has a width of o w which is equal to M minus w plus 2 p w divided by S w where S w is the stride along the width, w is the width of the convolution kernel p w is the pad applied along the dimension of width. So, each side you will be having p w and p w number of 0s padded down. So, that makes it 2 p w and M is the original size of the image before padding.

So, after padding the size of the image becomes M plus 2 p w basically. So, when and similarly for your height you have this sort of a relationship as well coming down. So, that is that is how your final modified thing comes into it. Now, the point is let us get into a practical demonstration of how it works down.

(Refer Slide Time: 05:34)



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	2	1	2	0
0	1	1	0	1	0	0
0	1	2	0	1	0	0
0	2	2	2	2	0	0
0	2	0	2	2	2	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	0	1	2	0
0	2	1	2	1	0	0
0	1	0	2	0	1	0
0	0	0	0	2	0	0
0	0	0	0	0	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

0	0	-1
-1	1	0
-1	-1	1

w0[:,:,1]

0	-1	0
0	0	-1
-1	1	1

w0[:,:,2]

1	0	0
1	1	-1
0	0	-1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	-1	-1
1	1	0
-1	0	0

w1[:,:,1]

1	1	-1
-1	0	-1
0	-1	0

w1[:,:,2]

0	1	-1
1	1	-1
1	-1	1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

4	5	2
-5	-6	1
2	3	4

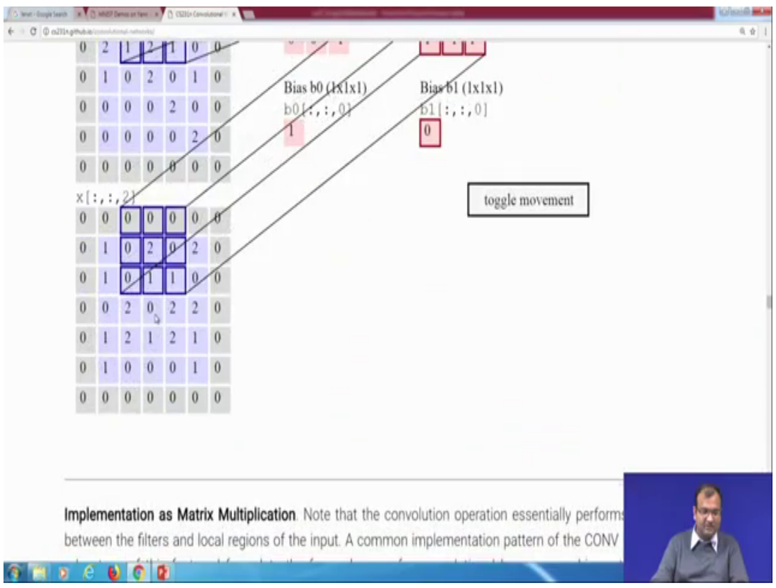
o[:,:,1]

-3	-1	4
0	4	5
-2	-5	8

So, I am using this example from well known course material from Stanford and, I chose to instead of redrawing the whole thing, I just chose to actually show it to you using just this animation over here, keeping in mind that this is one of the best animations to ever be able to understand it.

So, let us get into how it works out. So, you see this gray over here and these grayes are all the 0 padded 1s and this once in the blue, this is the 1 which is a size of the original image itself. So, this is a matrix part of the original image. Now my point is that I want the 3 cross 3 convolution kernel to convolved around this image. So, this side if you look into it this is my input image side over there ok. So, these guys write it down in a convention of 7 cross 7 cross 3, where 3 is the number of channels. So, this is my first channel, this is my second channel and, this is my third channel.

(Refer Slide Time: 06:25)



Implementation as Matrix Multiplication. Note that the convolution operation essentially perform
between the filters and local regions of the input. A common implementation pattern of the CONV

So, as in with the RGB image you have 3 channels. So, here I get the same thing coming down as well now, that I have these ones the next point is that I need to have a convolution kernel over here and, they also take down 2 different kernels 1 of them is w 0 the other 1 is w 1. Now w 0 has a 3 cross 3 cross 3 form, which means that it has a spatial span of 3 cross 3 and there are 3 side channels over there. So, the first handle corresponds to this first channel of the image, the second can of the colonel corresponds to the second channel and third channel corresponds to the third channel.

So, now if you see down these as some arbitrary weights associated over here, for the convolution kernel. So, this is on the 1st channel of the 1st convolution kernel. This is on the 2nd channel of the 1st convolution kernel, this is on the 3rd channel of the 1st convolution kernel and on top of that you have a bias as well, which is associated with each of these channels.

So, the first channel will have a bias of it is own the second channel has a bias of bias of it is own. Now when say it was starting to operate on this one, then yeah yeah it does get a bit annoying at laughter times that it keeps on moving while I am referring to. So, if you look into this animation. So, the first channel before of the kernel operates on the first channel of the image itself ok. The second channel of the kernel operates on the second channel of the image; the third channel of the kernel operates over here.

Now, accordingly this w 0 S they will give an output of o channel 0. So, when you see this 1 operating over here the total output is what is coming down on this channel, when you see this move down and go to this one the output keeps on coming on this channel over here and, that is how it operates. So, by our definition we had that the weight is multiplied by this one. So, here when I put down a 3 cross 3 matrix over here with this one, then I get a 0 into 0 is a 0 again a 0 into 0 is a 0 minus 1 into 0 is a minus 1 that is the first non ze minus 1 into 0 is again a 0 this, 0 into minus 1 is a 0 this 0 into minus 1 is a 0.

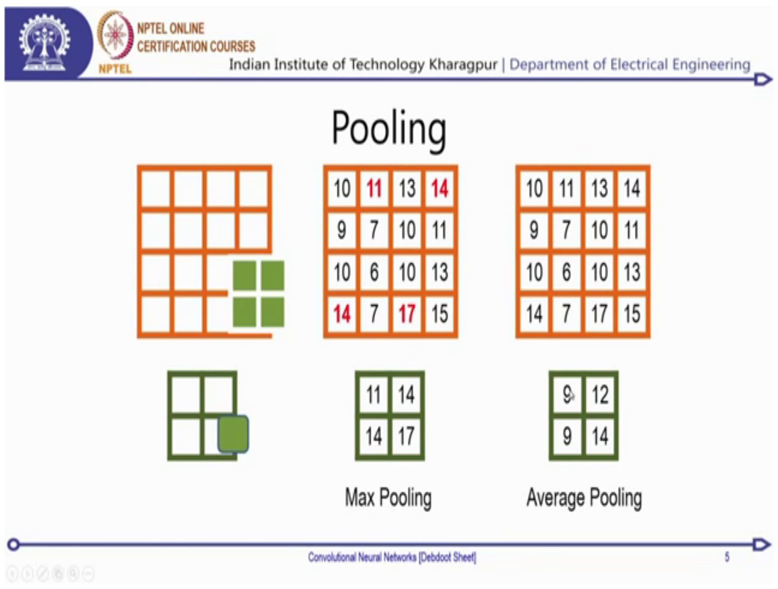
Now, this 1 times a 0 is 0 0 times a 0 is 0, this minus 1 over here is what corresponds to this 1 over here and, that becomes as minus 1 and this 1 and 1 dot product will give it to a value of 1. So, I have a minus 1 plus 1 which makes it 0, I do the same thing over here. So, over here when I am doing the same thing these 4 elements over here are all multiplied by these 5 it elements are multiplied by 0. So, you do not have an resultant option this 0 and this 2 will correspond. So, this 0 into this 2 makes it a 0 this minus 1 corresponds to this 1. So, there is a minus 1, then there is a 2 and then there is a 1. So, this effective result from this channels multiplication is ok.

So, I have a 0 from here and a 2 from here that makes it 2 and, now I take this particular kernel, I have all of these 0s with these elements over here. So, these 5 elements do not give me any results. Now 1 into 1 makes it 1 minus 1 into 0 is a 0, then 0 into 1 is a 0 minus 1 into 0 is a 0. So, the only result over here is a 1. So, I have a 2 plus 1 and then I take another 1 over here, which is my bias. So, 2 plus 1 plus 1 makes it 4 and that is the resultant value which comes down over here. So, you get down the point that it is very much a volume which is getting convolved and that is the result in which you get down over here.

Similarly, you can keep on doing for all other locations and you would see this 1 appropriately populated, you repeat the same thing for the 2nd kernel and, then you would be getting down the 2nd channel of my output coming down, but remember that the bias for the second kernel is what is present over here and, each kernel has it is own bias and that goes down pretty much with the yeah. So, this was the easy option of stopping at which I overlooked. So, if like we can go down to at any position. So, yeah say this is the second position where I am located at. So, now, you that you have stopped it down so now, you can again take the same kind of a multiplicative product sum all of the dot products add down the bias and, you will be getting this value of 5 over there.

Now, keeping 1 thing in mind that this is how a standard convolution would work. Now if you keep on repeating say these 2 outputs over here the 2 output channels and, then they are also exposed to some sort of a convolution, then you would have the same thing repeating over and over again and, then that is a pretty standard way in which this would be going on. So, with that let us get back into our convolutions as well. So, this is what we finished off discussing. So, that is how it was getting filled up completely and you had your height and weight. The next part was on understanding pooling operators.

(Refer Slide Time: 11:15)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Pooling

10	11	13	14
9	7	10	11
10	6	10	13
14	7	17	15

11	14
14	17

Max Pooling

10	11	13	14
9	7	10	11
10	6	10	13
14	7	17	15

9	12
9	14

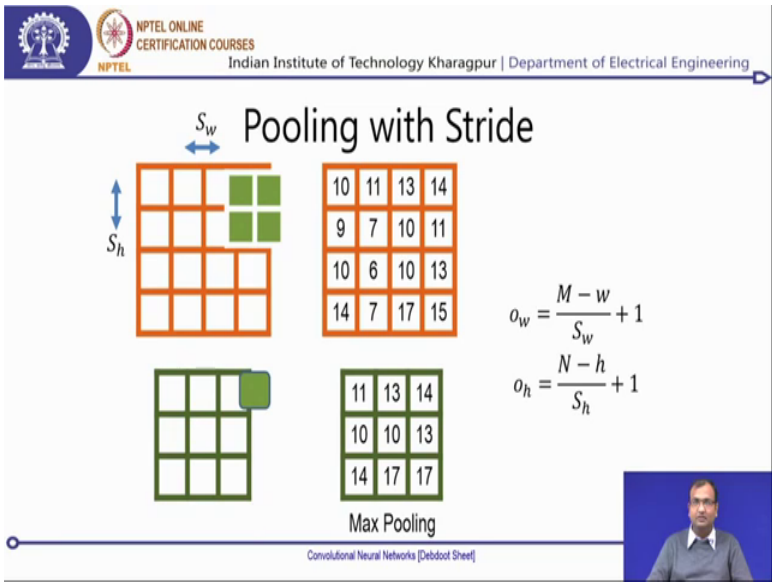
Average Pooling

Convolutional Neural Networks [Debdoot Sheet] 5

And these pooling was basically a way of reducing the size of an image. So, the whole concept which I said was that there can be an option in which I would like to reduce, this image size to half of it ok. So, the point is that if I have a 4 cross 4, then if I am reducing it on each dimensions to half of it becomes 1 4th of the total area and, get is converted into a 2 cross 2 matrix over there and, this is how it would keep on repeating itself.

Now this pooling can be done in terms of something called as a max pooling, in which what I would do is I would take down the maximum value and then place it in the middle over there, or this can also be done in terms of an average pooling, in which the idea is that in this 2 cross 2 matrix, I will take what is the average value and just resubstitute it over there in my pooling layer over.

(Refer Slide Time: 12:00)


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

Pooling with Stride

10	11	13	14
9	7	10	11
10	6	10	13
14	7	17	15

11	13	14
10	10	13
14	17	17

Max Pooling

$$o_w = \frac{M-w}{S_w} + 1$$

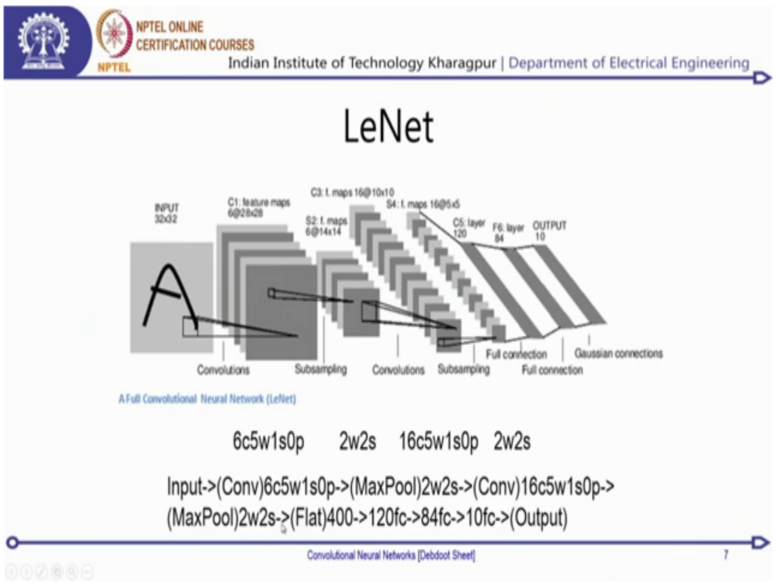
$$o_h = \frac{N-h}{S_h} + 1$$

Convolutional Neural Networks [Debdoot Sheet]

Now, if I want to do pooling the earlier case was where I was using a stride of 2 2 pool. So, that meant that my stride is equal to my width or an or height of the pooling kernel which I am using, instead of that I can even choose to do continuous stride and that makes it an interesting proposition over here. So, technically what this means is that if I have some sort of an average pooling going down over here. So, that is equal to some form which comes down over here and, this is my relationship between my width and height and, then this effectively gives me basically a dilated a grayscale dilated version of the whole image which comes down.

So, you can see quite amount of similarity between what is happening down at certain number of operators in convolutional neural networks, to your standard operators in case of a image processing based solution, in a standard classical computer vision based approach and these are the ones which work out.

(Refer Slide Time: 13:00)



Indian Institute of Technology Kharagpur | Department of Electrical Engineering

LeNet

A Full Convolutional Neural Network (LeNet)

6c5w1s0p 2w2s 16c5w1s0p 2w2s

Input->(Conv)6c5w1s0p->(MaxPool)2w2s->(Conv)16c5w1s0p->
(MaxPool)2w2s->(Flat)400->120fc->84fc->10fc->(Output)

Convolutional Neural Networks [Debdoot Sheet] 7

So, with that let us get into what is called as a LeNet. So, if you go by the standard definition this is the architecture which I have directly taken down from the authors paper and, I give full citation to them I have I have not redrawn the whole thing because, this is from a tutorial perspective this is possibly best rendering which we have for LeNet available.

Now, the whole concept over here is that you take an image of 32 cross 32 and this is a grayscale image ok. So, although we were doing our experiments on 28 cross 28 and the whole thing was basically by reducing 4 peripheral pixels over here to get it down into a 28 cross 28 and, then that is something which matches down these feature maps over here, but for reasons not quite well known to most of us we actually do not know what happened to this original which was proposed on the 32 plus 32 and, while the actual data which is available today is a 28 cross 28.

Now, how this goes on over here is that the first idea is take a convolution from this 32 plus 32 and, then this produces a feature map of 6 channels over here 1 2 3 4 5 6, which you see over here as the number 6 in C 1 feature maps and, then the size x y size is 28 cross 28, now from there the next point is that we do a max pooling and convert it to 14 cross fourteen; that means, that there is a 2 cross 2 and a stride of 2 max pooling which is applied and since there are 6 number 2 channels the same number of channels get preserved over there.

The next 1 is with C 3 which is the next convolution. So, this 1 2 3 this is the layers along the depth and CS are for convolution layers S are for subsampling layers, which the authors of themselves kept on defining. So, the next 1 is from a 14 cross 14 you go down to a 10 cross 10 and 16 such maps over there from the 16 you do a subsampling you come down to a 5 cross 5 size from a 10 cross 10. So, that is also a 2 cross 2 sub sampling and, then you have 16 of these. So, you have 16 into 25 number of neurons in total. So, 16 into 25 would make it basically 400 neurons.

Now, from 400 neurons you map it down through a fully connected day layer 100 and 20 neurons, which is C 5 from there it maps down to a fully connected network with 84 neurons from there it maps down to a fully connected network with 10 neurons and over here, what is called as a Gaussian connection is basically a sigmoid transfer function. Now with this kind of a sigmoid transfer function you can actually stretch down your values in the range of 0 to 1.

So, typically since this is a digits classification problem over here on the output side there is a 10 cross 1 neuron bank and this is a 1 hot neuron. So, any 1 of these neurons is going to be 1 the rest of the neurons are going to be 0. So, we have a convention actually of writing this one and the idea of writing this one is something of this sort. So, if we try to write down a convolution we would try to express it in something of this one.

So, 6 c means the number of channels in the output, which we need to produce and that will basically define the number of such convolution kernels, which I need to have within my convolution operator, 5 w over here means I am using a 5 cross 5 kernel size. So, this is width of the convolution kernel which I am using and for simplicity we try to use square kernels, which means that the width and the height are same, there are possibilities of using say rectangular counters as well and within your coding exercises you would see that we always put down. And so, when we start down with the convolutional ones you would see that we put down 2 dimensions over there. So, the first dimension is the width and the second dimension is the height of the kernel.

Now, there is pretty much an option that you can put both of them the same, which works out pretty much for natural images, but then there are certain sort of images in and say which are non-linearly sampled. So, your x sampling along the x direction and sampling along the y direction is pretty different, in that case you would more of preferred to put down 2 different kernel dimensions.

So, your width of the kernel would be very much different from the height of the kernel. Now here what I do is I take a stride of 1 and, then a padding of 0. Now one question comes down is I have already written this 1 down for you, but can you actually given this information over here can you actually find it out. So, that that is a very pretty exercise question which you can a lot of find x expect.

So, it is not just from a examination point of view that you big ask this question over there, but then most of the authors and then most of these publications which have these convolutional neural networks, they would just be drying out the network to you, they would give down dimensions of the results at each of these layers, but most of the time not even speak about what was the kind of the convolution operator which was used over there. And, now you have to figure out from your side that what can be an optimal combination of these convolution kernels which were typically used. So, we generally start by configuration whereby if we take down the minimum size of the convolution kernel and, here the idea was that if we if we go down by the concept of a minimum size of the convolution kernel, then let us see what comes down.

So, you see that the size was decreased on both the sides by a dimension of 4. Now if that has to work out 1 option maybe that I do not put down a padding over there and, then I take a 5 cross 5 and take a stride of 1 and, that pretty much fits down within this kind of a constant yes you can have other convolution options as well and, then pretty much I would say that please try them out as well to see which other sort of a combination, but 6 c in (Refer Time: 18:51) will remain constant because your output over here has 6 channels, unlike anything else and that is pretty much something which defines the number of channels over here.

So, similarly let us look into this subsampling over here, now the subsampling if you see. So, it was going down by a factor of 2 on both the sides. So, that and the number of channels will be preserved because that does not have any impact on subsampling. So, this would mean that I have some sort of a 2 cross 2 kernels and a stride of to employ over there. So, this subsampling layer over there or the max pooling is with the 2 w 2 S ok. The next 1 is again a convolution which produces 16 channels output we again use a padding of phi of width of 5 kernel over here, stride of 1 and a 0 padding over there and, that technically brings down 14 grows 14 channels folding cross 14 sized output to a 10 crossed and sized output.

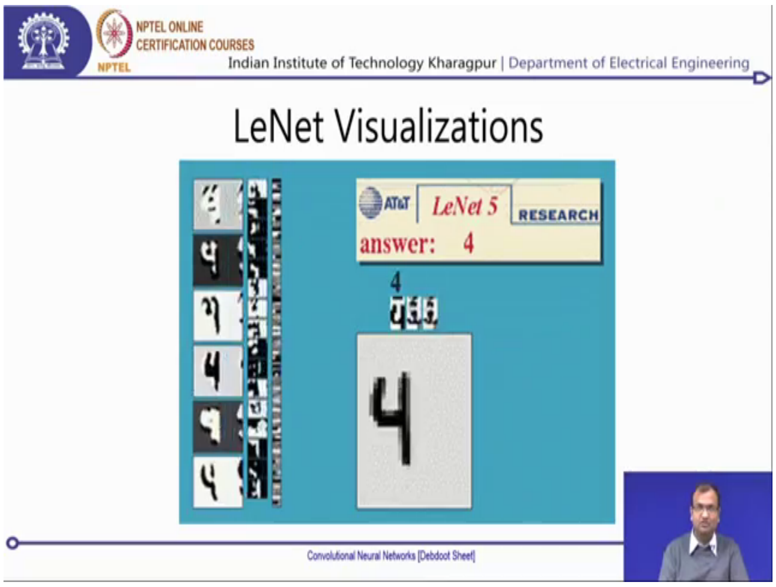
So, remember it was w minus so, so your this dimension M minus w which was the width of the kernel plus 1. So, that makes it plus 2 p w; obviously, now your pad over here is 0. So, that does not contribute you are width over here is 5. So, it becomes 14 minus 5 plus 1 and that makes it 10 and, that divided by 2 S which was your stride over there and the stride since we are keeping ah. So, divided by S w sorry. So, S w was your stride and since stride is 1 so, this whole division factor over there unifies and then you get down your option.

So, similarly the next sub sampling and the max pooling is of 2 w 2 S and the whole network can accordingly be represented something like this that, you have your input which comes down from your input you do a convolution with 6 c 5 w 1 s 0 p ok, from there you get into a max pooling with 2 w 2 S from there you enter into a convolution with 16 c 5 w 1 s 0 p, from there again you have a max pooling of 2 w 2 S, from there you have a flattening.

So, this flattening is to get down 400 neurons which has which comes down over here. So, you linearize and get down a flattened layer of 400 neurons. So, which basically means that here do you have 16 into 5 into 5 that is a 3 d array in which a top level flattening is an operator which basically converts this 3 d array which has 100 such points into 1 single 1 dimensional array. So, it is a 400 cross 1 array which you have as of now and, that can be now connected down to your fully connected layers.

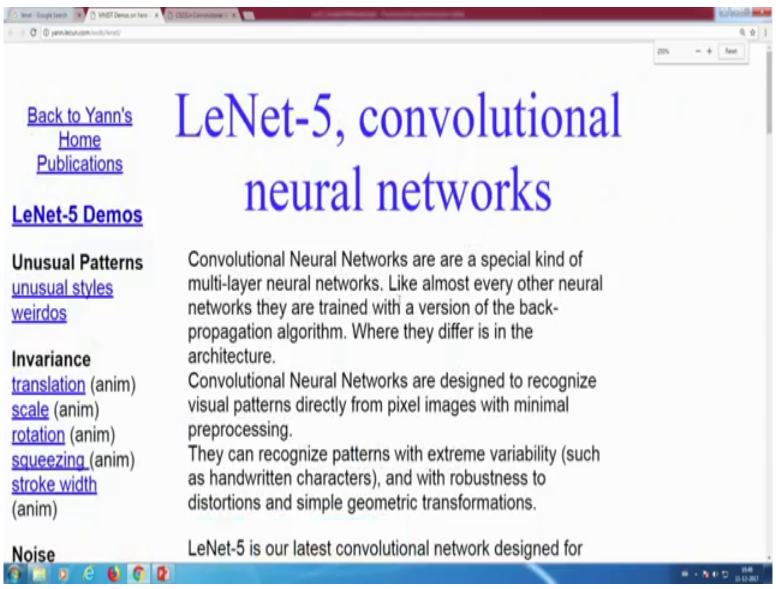
So, there is no learnable parameter as such when you are flattening it out it just some sort of a indexed arrangement into 1 single 1 d array, that 1 d array can now be connected 100 and 20 element array through a fully connected layer and these weights over here are trainable weights, from 120 they go up to 84 these are again trainable weights from 84 it goes again to the output, these are again trainable weights and here you have a nonlinearity of a Gaussian connection. So, this together defines my sort of connectivity throughout the whole network which I have now.

(Refer Slide Time: 21:58)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
LeNet Visualizations
AT&T LeNet 5 RESEARCH
answer: 4
Convolutional Neural Networks [Debdoot Sheet]

So, how it works and looks like is something of this sort, so I have taken down this from the direct website of LeNet and, you can actually get down much more details if you are there on this particular website of Yann Lecun.

(Refer Slide Time: 22:11)



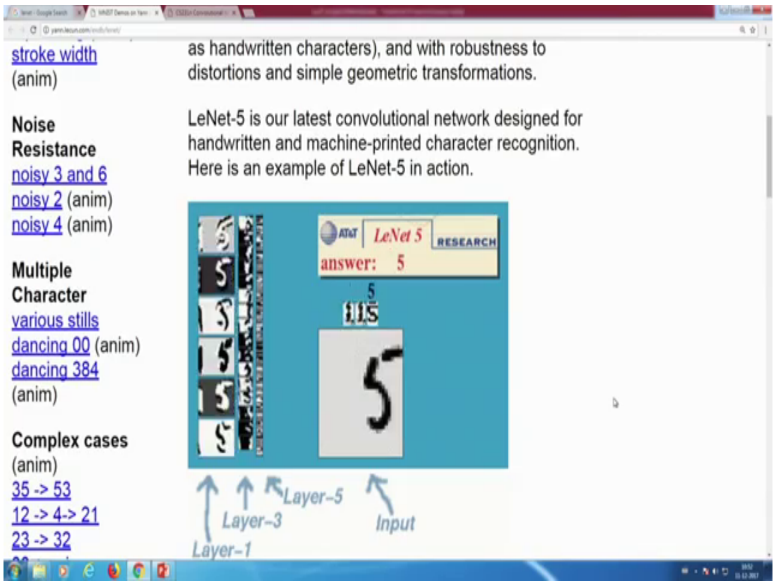
LeNet-5, convolutional neural networks

Convolutional Neural Networks are are a special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm. Where they differ is in the architecture.
Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.
They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.

LeNet-5 is our latest convolutional network designed for

So, these are from the Yann Lecuns website and let us just zoom into that. So, if you look over here.

(Refer Slide Time: 22:19)



stroke width
(anim)

Noise Resistance
noisy 3 and 6
noisy 2 (anim)
noisy 4 (anim)

Multiple Character
various stills
dancing 00 (anim)
dancing 384
(anim)

Complex cases
(anim)
35 -> 53
12 -> 4-> 21
23 -> 32

as handwritten characters), and with robustness to distortions and simple geometric transformations.

LeNet-5 is our latest convolutional network designed for handwritten and machine-printed character recognition. Here is an example of LeNet-5 in action.

LeNet 5. So, LeNet 5 comes down from the fact that you have total 5 number of layers over there. So, he has a very good implementation of so, if you recall back from our discussion over here and we count down the number of layers. So, there is 1 2 3 4 and then this whole part over here which becomes 5, or another explanation is basically taking down the number of flow interval parameters. So, you have 1 convolution which has 1 learnable parameter 1 the second convolution, which is a learnable parameter 2 because subsampling does not learn anything.

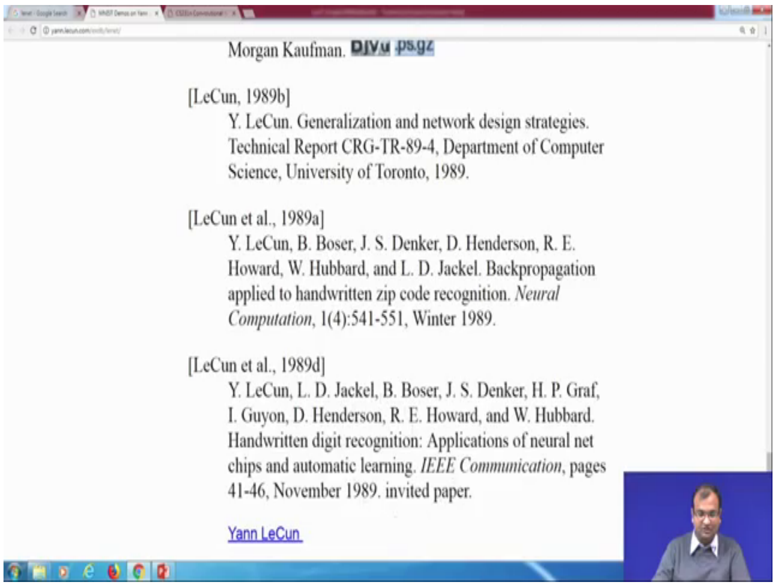
Now, then so we have 2 layers which have 2 different learnable parameters, then from here you get down to fully connected layers. So, 1 2 3 there are 3 weights which you need to learn down over here, in total there are 5 different non-linear transformations and learnable parameters through which this network works and for that reason this is also called as a LeNet 5 layer.

So, here if you look into it so, he gives a her an intuitive explanation of layer 1. So, this is basically the output which comes down from the first convolution result. So, you remember that there were 6 such channels. So, this is 1 2 3 4 5 6. So, you have an image this input going down over there you convolve with 6 different kernels and, this is the output you see on each of these resultant over there. Now from there it goes on to a layer 3. So, layer 2 was basically a subsampling layer. So, that made it half of this size. So, from a 32 cross 32 it became 28 cross 28 from here it became 14 cross 14 and, then we had 16 such combinations coming down.

So, you have 1 2 3 4 you can just count on there are basically 16 such layers over there in layer 3 and, you had another sub sampling which make it made it to 5 cross 5 and, then we had a convolution run down on that 1 as well and this is the output of that. So, there were total of 16 5 cross 5 channels which comes down over here. So, this is the result which comes down. Now if you see in the initial you had an image, then you had some sort of age oriented or accentuated versions of it, then we were getting down some sort of a sparse representation and finally, what you get down towards almost the end which is your 400 neurons this is technically a bit stream.

Now, this big stream is something which is important in terms of firing out your neurons to classify, what is being seen over here and, that is how this result keeps on coming down. So, what they do is that there is a running window kind of a thing. So, you start from here and then here and you have 3 different detections being carried out and that is what is being shown over here now if all the 3 detections when they populate and show the same number, you see that number also coming down over here as the answer.

(Refer Slide Time: 25:09)


Morgan Kaufman. DjVu ps.gz

[LeCun, 1989b]
Y. LeCun. Generalization and network design strategies. Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto, 1989.

[LeCun et al., 1989a]
Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541-551, Winter 1989.

[LeCun et al., 1989d]
Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. *IEEE Communication*, pages 41-46, November 1989. invited paper.

Yann LeCun

So, this is based on a lot of papers and work which they had and the earlier ones was what came down around in 1989. And since then it has been going on. So, there they have been lot of advances put down onto this particular field and, till recently when we had the boom. So, over a long period of time it was really hard to actually understand and create down these bits and create down a learnable aspect over there and, accordingly to took a lot of efforts for the community to grow up, but today that we have access to compute power. And we have a good amount of access to a libraries and making out easy prototypes for these methods, they are coming down into the forefront.

So, in the next lecture we would actually be implementing LeNet 5 by ourselves, you writing down our codes and doing it and, eventually from LeNet 5 we will be getting into understanding more and more details of how other convolutional neural networks work and, what modifications can we do within that architecture. So, what can be different kind of so, as of now we have just studied on the architecture, but you know that there are other aspects of cost functions, there are aspects of learning ability your learning rules, your optimizations, how do you handle down batches and do it.

So, we will be using them as a lot of different examples in order to come down of how a practical network can be trained. So, till then we just stay tuned and then get back to you later on.