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

Lecture – 31
Convolutional Autoencoder and Deep CNN

So, welcome, and today we will be starting down with next portion of it. So, while we have studied down a few of the standard techniques for multilayer perceptron initialization using auto encoders till now, and then we have also stepped into the basic blocks of convolutional neural networks, and till the last class, we have been doing a very simple and the first kind of convolutional neural network, which was the linnet which was available.

Now, today what we are going to do is, something called as a deep CNN, and I would be discussing on the very basic versions of deep convolutional neural network which came down. And eventually after this so, in the next few classes we will be doing our coding exercises which are related to these kind of convolutions.

So, the first one which I would be starting down is a convolutional auto encoder. So, this goes down with the same philosophy, as when you had with a fully connected auto encoder, except for the fact that in case of a convolution of water encoder, you have this advantage that we do not have fully connected neurons any further, but now all these connections are convolutional.

So, that does give us a very distinct advantage over any of the earlier methods in the fact that while in a standard auto encoder, where you had to all neurons fully connected down. In case of a convolution, because these are convolutional connections going down;so, now, you pretty much have this comfort available that you can have space invariance as well coming up over there. Now with this distinct advantage coming to play with convolutions over there, there are certain distinct advantages in terms of what mode you can do. So, now, you can build up on any of the earlier versions of auto encoder which you have studied.

So, including stats parts (Refer Time: 01:59) auto encoders and instead of having fully connected neurons over there, you can now have convolutional connections established between the neurons and this would help you gain down spatial invariance, which is a very important factor, and as well as random learn much more compressed and compact representations which can be convolutionally expressed, in order to make more meaning coming out of the data. So, today without much of a delay let us get into the topic. So, what we would be covering is, something of this sort.

(Refer Slide Time: 02:31)

NPTEL ONLINE CERTIFICATION COURSES

NPTEL

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Organization

- Convolutional AE
- AlexNet for Image Classification

Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 2

So, initially I would be discussing convolutional auto encoders and then I would book down to one of the image classification problems with the particular network called as AlexNet. And this is used for solving one of the best problems in the field and that is called as the image net classification challenge.

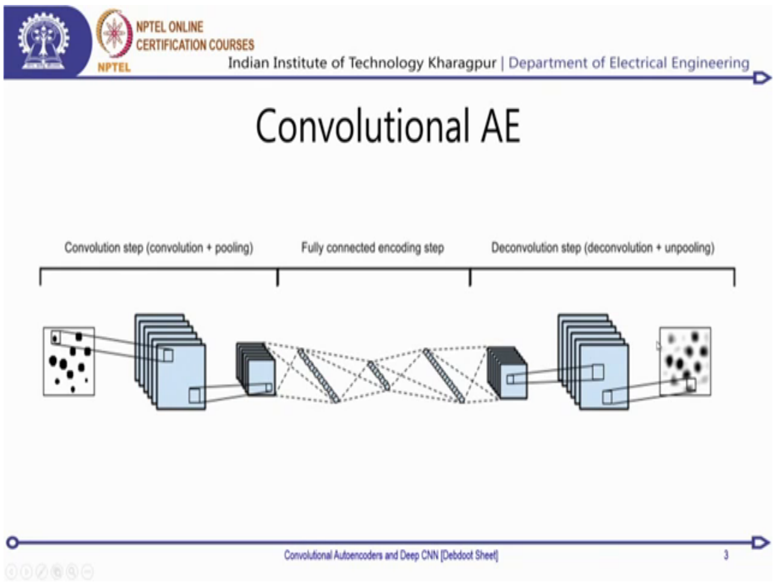
So, image net is basically a challenge, where the idea was that you have about a million images given down to you, and then there are about 1000 categories in which these images can be divided. So, the whole idea was that, can we actually divide them like train a classifier in order to do a 1000 classification problem in the best possible way, on this million images. And the next one is VGG Net or which came down from the visual geometric group at Oxford and so, they just named it as VGG from their own group.

So, so, these two over there AlexNet and VGG Net are some of these early precursors of the most stable kind of architectures which came down in order to solve this problem of classification. So, giving you a brief; I would be giving a very brief preview of what the image net challenge was. So starting in, so, 2017 basically saw the sort of what we call as the ultimate solution of image net.

So, there is not any for further research and activity going down on image net solutions, because what we felt out in the community is that, whatever could be done in terms of understanding images and classifying images has been done. We had to the extent of having like really good convolutional neural networks, really good, very deep neural networks, not necessarily only convolutional, which can solve out this problem of image net classification, and subsequently the idea is to move over to even further better kind of ways.

So, more complicated challenges say, can be identified actions in video or what today is called as the video activity classification problem or activity net challenge itself. So, that is something, which is now one of the most predominant driving force within the community and is an active research problem. But nonetheless since we are doing a classroom level course over here; so, until and unless we are done with images it does not make sense for us to even move to videos. Though, that is also there in the syllabi of this particular course which we are doing.

(Refer Slide Time: 04:45)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Convolutional AE

Convolution step (convolution + pooling)
Fully connected encoding step
Deconvolution step (deconvolution + unpooling)

Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 3

So, without much delay let us get into this one. So, convolutional rotary encoder is typically something which is represented like this. So, here the idea is that initially you have your inputs given down over here, and the whole objective is that as in standard auto encoder you would like to reconstruct the same as an output over here.

So, we are doing the same thing over here, except for the fact that you would see that it is not a perfectly reconstructed output, but there are some blowers and hazy things which come down over here, and that is pretty ok, because if you are just doing, getting the exact one estimate, it does mean that you are at the best of the accuracy, but that also has this risk that you have possibly over trained, and let your network memorize patterns over there. So, that it’s just a given a test of time. So, the more you train the better it becomes over time, the more the data you expose it to, the better validation schemes you have it works out pretty fine.

Now, here what comes down is that, you have your original image over there, and then the idea is that you will have a set of filter banks. So, a set of kernels and so, what we see over here is that 1 2 3 4 5 6. So, you remember as in you had for a linette where you had just 6 corners over there, and then you convolve it with this original image would get down 6 such outputs coming down. So, they are these outputs come down. Now, from there you do a sub sampling and this is where you would each dump. Now, once this sub sample layer is present over here, then this part is the fully connected. So, as in with a classification problem, as in the linnet, the objective was that once you went down to the last layer where you had 16 cross 5 cross 5 and you got down 400 neurons over there, then you did a linearization of these 400 neurons and then these linear neurons where this this sort of like 400 cross 1 array of neurons, tensor of neurons was connected down to 120 neurons in the next layer.

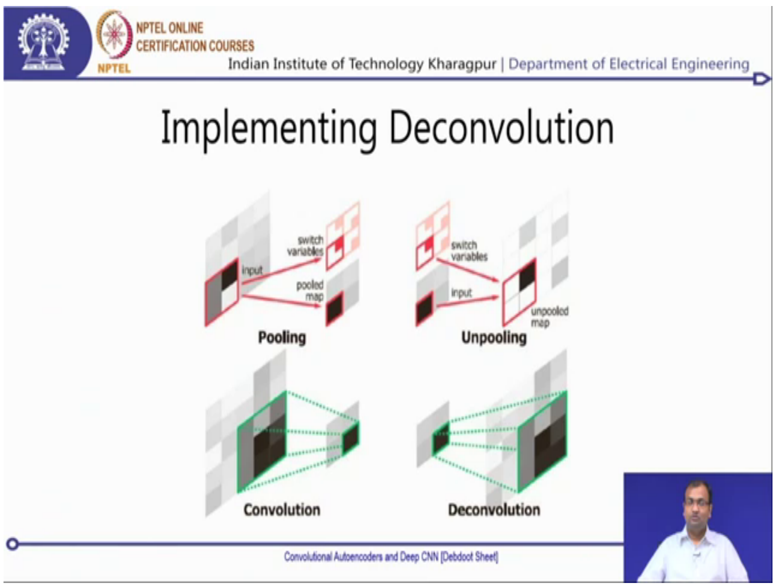
So, here its almost of a similar thing, except for the fact that over there it was a classification problem. So, you connected on 400 220 from there to 84 and from there to 10 and then you just start a classification output coming down, but here since we are interested more of in constructing a auto encoder so, we go down to a lower space, but from there we start expanding up as in with the fully connected auto encoder. So, with the fully connector auto encoder you just had this input step, you had your hidden layer and then this output coming down and you wanted to match this one to that one. Here what goes on more is that, once you have these linear neurons available over there, you will try to rearrange them, repack them and to get down a convolutional volume.

Such that you can convolve and get down this second of the volume and from there you can again convolve and get down this one. the only difference would be over here to here, so this is an up sampling which you had contra to or down sampling over here and this step is often called as un pulling operation or a deconvolution operation in itself. So, as, if you remember clearly while I was explaining the initial blocks of a convolutional neural network.

So, I did explain about what is a convolution? What is a pooling operation? What is a sub sampling operation, as well as, then we went on to another stage, where it was called as a deconvolution and then I made a specific point over there that we will be touching upon deconvolution a bit later on and an that is an interesting stage when it comes down. So, now, this is a stage when we really need to speak about the deconvolution operator. So, here what we do is, that you have your input present over here, but then I need to go to this stage, which is an sort of a magnified version of it. So, now, one way of doing it is, trying to do a un pulling. So, within that un pulling the whole logic was that, sorry. So, within the un pulling the whole logic was that you would just be up sampling and replicating at out over there, but then you know that while you are replicating. So, if you have a 2 cross 2, you can make it double of the size and make it 4 cross 4.

So, every single pixel over there, every single location gets replicated to 4. So, a 2 by 2 small matrix over there, now that this replication is happening, so you will be getting down a major problem of what is called as a blocking artifact or a patchy light effect. Now you have to get done much smoother, one option may be that you can do a bilinear interpolation and solve it out, but instead of that, one way is that you convolve and interpolate it up over there and that something which is called as a deconvolution, which we will look in the next step. from here to this image space is again as very standard way. So, you just convolved and then get your output over there, there is just one convolution kernel which will be present from going down from these 6 channels on to this one single channel over there, and that is what we can define down the network pretty easily and without any kind of a problem.

(Refer Slide Time: 09:23)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Implementing Deconvolution

Convolutional Autoencoders and Deep CNN [Debdoot Sheet]

.

So, deconvolution is this where the main challenge comes down. So, you see that if you have your sort of a pooling and then if you would like to do a corresponding thing, which is called as a unpooling. So, over there how its related down is that; say I have my 2 cross

2 patch over there, which is also called as a swatch. So, and from this patch what I can do is, I would like to get done what is my resultant pulled out versions. So, this resultant pulled out version over here is what is called as the pool map. and now I see that this black patch is what is the resultant of some convolution of this pooling. Now that I got this one for this small 2 cross 2 patch getting represented into a one single pixel location over here. I can also select down what are my variables which actually led to that. So what this would help me is that when I am doing the other way around which is going from this smaller version to the bigger version which happens over here called as unpooling. So, this is my smaller version, from here I am going to the bigger version. So, I know where exactly this had resultant. So, I knew that this black patch actually came from here.

So, if I am going to replace it into my unpooled version over there. So, we are putting my black patch exactly at that location instead of going through any other of this place. Now that I have placed this whole thing over there the major challenge which comes in that I have some sort of a sparse representation, because the rest of the places I do not know exactly what was there that was not stored with me. I just had the macs pulled out value and the location from where this macs pulling was happening, which I can go and replace it back over there in the unpooling time over there.

Now coming with the same logic of a deconvolution what happens is, that in a convolution, you have done a weighted, sort of weighted sum and then you have just replaced it at this location. So, this was a convolution with the stride which was happening. Now if I want to do a deconvolution. So, I will have to have some sort of something called as a fractional stride as well, because if I do a stride of say 1 then I get the same size of an image in a convolution. if I do a stride of 2, I get half the size of the image in a convolution.

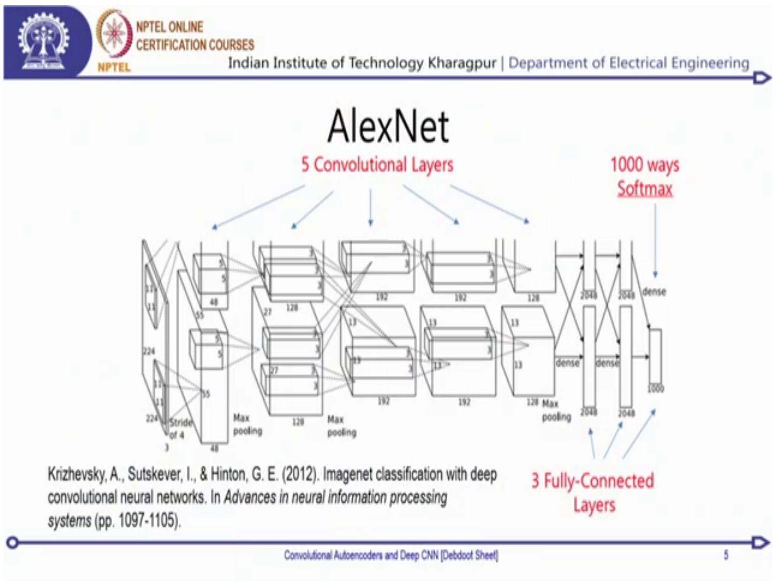
Now in the same way if I do a stride of 0.5, I should be getting down the double of the size over there. But then you cannot technically imagine like what will be your double the stride and that becomes really problematic. So, the deconvolution block is something which works on that way with which does a fractional slide. So, there has been a new theory a very recent theory on something which is called as deconvolution, intuitively. what deconvolution is trying to do over there is that, deconvolution is a way of unpooling all of these indices. So, the idea is that using this version over here, if I have multiple of these.

So, if there are say 4 such unpooled maps for convolutional layers which I can generate out over there, and then shift and translate and freeze them onto this bigger patch, then I should be able to get down a bigger version of the whole image and that is what comes down and what has been found out from the theory. So, I will put down pointers onto reading down more details over there, except for the fact that deconvolution is not something to be discussed at much of a detail over here.

So, when we are trying to I mean at this particular lecture stage basically. So, on a little bit of lecture stages when we would be much more well versed with how convolutional neural networks are working, how convolution auto encoders work and get into transfer learning and computational complexity. At that point of analysis I would be bringing in back the concept of a deconvolution and actually trying to solve out the whole math concerned around with it. So, for now what you can actually take as a summary, just to be done more detailed in later part as a deconvolution is something which solves the opposite of the problem.

So, in convolution if you are involving and coming down to a lower channel with a lower size deconvolution is just an other way around, which has a result production which is much more conformal then what would be getting done with just a nearest neighbor interpolation or a bi linear interpolation; that is a summary and just of a deconvolution, but then this plays a significant role when you are coming down over here in on two encoders where you have to go down to bigger swatch as well. So, this particular blocks do make a very important statement in terms of their criticality of hues.

(Refer Slide Time: 13:48)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

AlexNet

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 5

So, having done the convolutional auto encoders the next part is to get down on to what is called as an AlexNet. So, this is a version where, so this is the original version of AlexNet as was implemented in 2012 for the paper which was, a first paper with a deep neural network which was able to beat down any of the conventional and classically available models for solving the image net classification challenge. And here as it went down was that you had images which were of a much bigger and a standard size.

So, the image net problem itself mandated that all input images to it were 224 plus 224 cross 3. So, they were all color images. The x y dimension was 224 cross 224 as you see over here. Now what this network was doing is that, initially they take down convolution kernels of 11 cross 11, and then they do a stride of 4 and then create out this next version and there are such 96. So, you would see that there is a 48 bank over here.

So, there are 48 layers and there is another similar 148 over here. Now given the compute requirements and the GPU constraints, which they had at that particular point of time, what they decided was that it was not possible to have all 96 kernels evaluated out together, that is why they evaluated out 48 and then 48, it was just for the purpose of memory handling down in a much better way. Now as in essence what we would be implementing today. So not exactly in this episode, but in the next lecture we would be doing that. So, today in the terms of contemporary what we implement these days is that you can put down all 96 once over here, the next one is 128 and 128. So, that becomes 256 corners.

Now what do you see over here is that here the width and height of these convolution corners is 11 cross 11, whereas, in the next layer the convolution corners become 5 cross 5. Then in the next layer they again become 3 across 3, in the next layer they become again. So, they keep on remaining 3 cross 3 and go. Now if you clearly look over there, there are 5 convolution layers as has been pointed out over here and then it goes to a fully connected neuron model over them.

So, there it connects down to 4096, then again 4096 and from there to 1000 classes which are for your classification purposes, and this is a 1000 cross 1 1 hot vector which gets on the output side of it for the purpose of classification. Now clearly looking at it if you see. So, initially my kernels are quite bigger they are 11 cross 11 and then subsequently they become start becoming half of the size. So, about half of 11 cross 11 is 5.5 or what is the kernel size of 5.

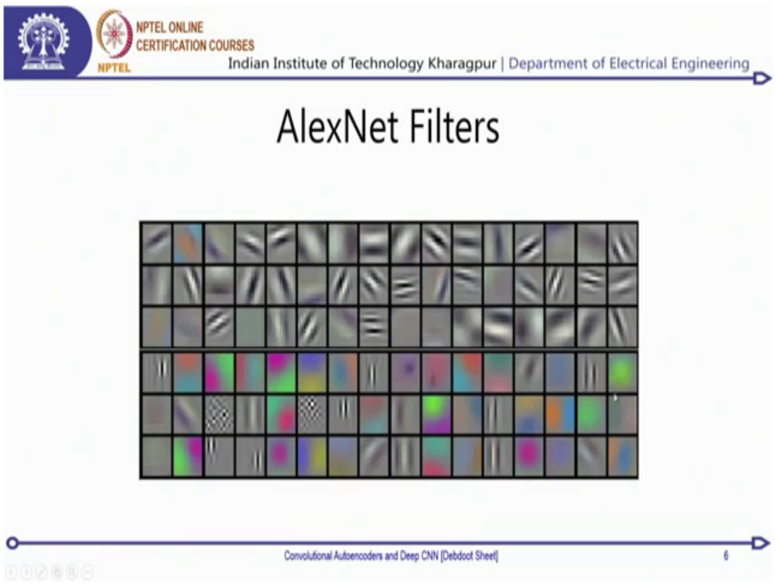
Now about almost half of 5 is 2.5 and that is why they bring in the kernel size of 3. So, this builds up as if a pyramid which you would typically be building when you are doing a multi resolution image processing as well. So, AlexNet as such was just a clever way of doing a multi resolution image pyramid, whereby you do not have convolution kernels which for feature extraction which are empirically defined but all of these kernel weights were getting trained on the run as the network was learning itself to classify images.

So, my target was to classify image, I was not giving exactly based on what featured you would be classifying image. Now as the error back propagation goes through and you have these weights modifying over here. So, what in essence happens is, that these weights get modified; such that you are now able to classify images in the best possible way, and these kernels would be tuned down to become kernels of a wave which helped the maximum discrimination between objects of multiple categories, and that is what AlexNet in total as such.

So, you can get through this particular paper which is available as of now open source, because it was published in nips in 2012. So, all nips proceedings are pretty much available on the open place you do not have to worry much about from where to get them, and this is one of the best ways of going through it.

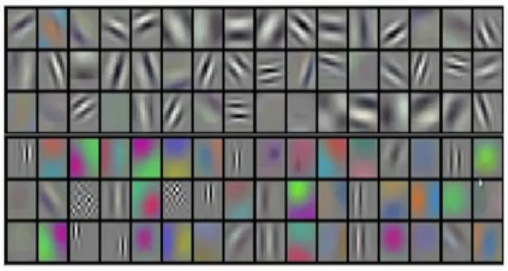
So, do make sure of going through AlexNet on this paper, because the assignment problems are not naturally going to be only from what is there on the slides there. There will be some interesting questions which would be getting down, for which you might have to go through the paper and that is possibly the best way of learning deep learning today. So, from the kernels over there, which I had also discussed in earlier as well showing you.

(Refer Slide Time: 18:14)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

AlexNet Filters



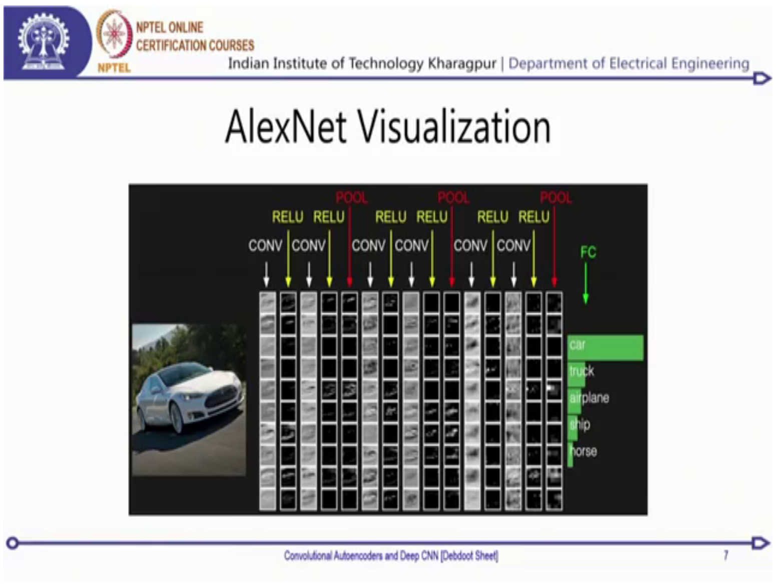
Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 6

This particular image, when trying to discuss about deep neural networks and deep CNNS coming now. So, these filters which you see over here, these are basically kernels. Now few of these kernels do replicate and become like almost gabber wavelet us and with a shifted version and rotated version, some of these become colors and the reason because in the original one you had a color image. So, you had this R G and B, three different channels corresponding to the whole volume of the kernel and then each getting visualized in itself.

So, some of them are say over here it's a red blob emphasis filter, this is a green to red edge emphasis filter, this is again a green blob filter, this again becomes a green blob filter. So, they do play a very significant role and if you look into these kind of filters, a checkerboard kind of a pattern. So, these are very high so, high frequency filters which come down, and maybe of all the 1000 categories which needed to be classified over there in image net. some of these objects where what needed these kind of very high frequency filters to come into play and that is why the year Scotland. the interesting factors that they do resemble a lot of these filters which we have designed classically till date and then that brings in the power of deep learning.

So, if a you have the whole model defined out pretty fine and as well you have a COS function which is defined your experimental data is enough, then you can pretty much using just the data itself and a cleverly crafted learning rule which we are doing these days cannot learn down patterns to extract features from images itself, and that solves a lot of the low level tasks which we would have otherwise done to solve (Refer Time: 19:51) signal problem, and now the whole art of visual computing comes to a much higher level, much significantly higher level, where you do not need to sit down and handcraft features any further.

(Refer Slide Time: 20:06)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

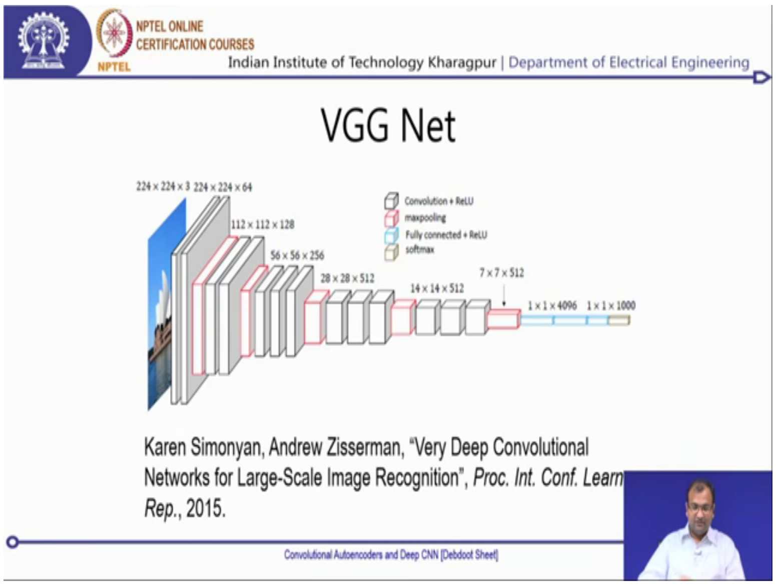
AlexNet Visualization

Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 7

So, if we go on and try to look into how this was working down. So, if you have an input image of a car into this network, then it would convolve take a rectified linear unit transformation function, then take a convolution then a really transfer function then do a pooling and eventually it would keep on going. Now as it keeps on going over here, now what you would observe is that, it's no more the original form factor of the car which is visible and then this just start becoming some sort of bit codes as you see over here, just blacks and whites coming down. and that is something which fires down the neurons and makes it a really good to identify with a very high likelihood that it is a car, with a low likelihood, significantly lower second likelihood of it being a track next a plane. So, I do not know exactly what it, figured out that it said a plane, but the probabilities again low and the whole aspect is that, when you are dealing with probabilities and you are just filtering it out.

So, even if the dynamic range between them is there, but still the top 5 predictions would just be coming up as it is, though their probabilities are very low, and the network is actually much intelligent in order to say that with the high efficacy it's actually a car, not any of these others.

(Refer Slide Time: 21:26)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

VGG Net

Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *Proc. Int. Conf. Learn Rep.*, 2015.

Convolutional Autoencoders and Deep CNN [Debdoot Sheet]

.

So, there ends up AlexNet and we can get into the next version which is called as a VGG

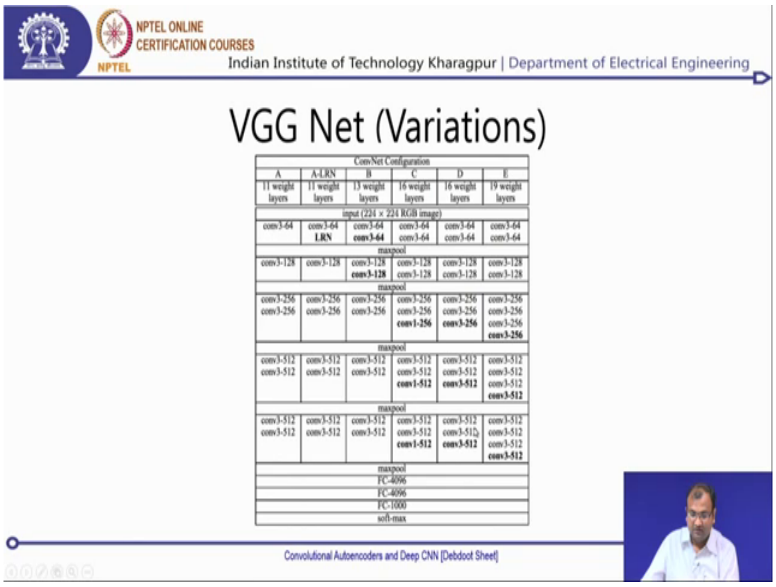
Net, and this was a much later on version, comparatively recent or what you can say, because it came out in 2015 and this whole thing had something interesting. So, if you look over here, there are sort of like banks in which convolution happen. So, if you look over here, so you have an image of 224 cross 224 cross and then it convulse and gives you two channels over there. So, there is one level of convolution which gives you 224 cross 224 cross 64, then again that conversion gives you 224 cross 224 cross 64. Now from there you have your max pooling for the first time coming over here.

Now after this max pooling, it again does two convolutions then does a max pooling, then does 3 convolution, then does a max pooling, then does 3 convolutions max pooling, then 3 convolutions then max pooling, then linearization and then a final decision coming down over here and that is how it was working out. So, here you can actually pretty much visualize the structure of the network, and how they put down is that this sort of length of this block along this x direction over here, that is equal to the number of channels present over there and this and this, so this is your x direction, this is your y direction and this is your z direction or along the z direction is your number of channels.

So, as the channels keeps on increasing the a width over therefore, each of these blocks increase and as the height, and so the x y dimensions of the image or the space which keeps on decreasing. So, you see this one. So, this pretty much looks like a pyramid. I mean hold it vertical and then you would see it or I mean till your screen or tilt your head you would be getting down a pyramidal like structure.

So, but it becomes really hard from here to understand actually, what is the nature of the convolution corners whether they took up padding over there, what is the kind of stripes which they were taking and then what is the kind of a max pooling operator, whether that was max pooling on an average pooling and what was the size of the kernels and the stride of the pooling operator as well. So, let us get into the details, because in this paper its a very nice way in which they have done.

(Refer Slide Time: 23:30)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

VGG Net (Variations)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Convolutional Autoencoders and Deep CNN [Debdoot Sheet]

.

So, they have not just presented one single a kind of a network over there, but they have put down multiple variants over there. So, one is a 11 layer network, the next one is also an 11 layer network with a modification called as LRN local response normalization, so, I will come down to that briefly in the next slide. there is another one which comes with 13 weight layers and 16 weight layers and then final 19 weight layers. now weight layers, which when they call over here.

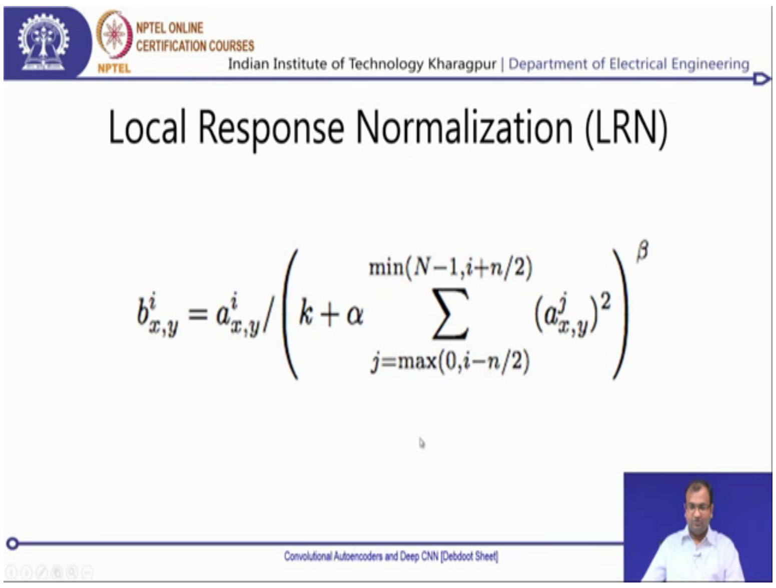
So, weight layers are basically layers, when you have certain number of weights to be learned. Now with a convolutional neural networks the weights are only just the convolution kernels which are to be learned, if you do a max pooling that is not a weighted operation. So, the network does not learn anything new over there, there are no parameters to be updated. So, weight layers are just the tunable parameters if they are present over there.

So, here initially, they have on this 11 1 convolution with so, this three basically means that they have a 3 cross 3 cornel over there A. So, the next one also has a 3 cross 3 control over here, the next one B, which has 13 layers, you see that there are two such 3 cross 3 convolution kernels and there are 64 channels in each of the and this is how this whole thing is defined, and wherever there is a max pooling, it is a maxpool and then from the purpose that it was always reducing it to half and looking more into the code part over there, he did realize.

So, when I go into the coding part I will show you exactly, if we import the original architecture proposed by the authors and how you can from there understand what was the math which was done over there. So, this was the 2 cross 2 pooling which kept on and eventually the rest of it is the same. So, it just depends on introducing more number of convolutional layers at every layer, and then since these are equal sized convolutions of 3 cross 3.

So, with the 3 cross 3 kernel, as we have done from the earlier one. So, you know that if you want to have the same size of an output then you would need a padding of one and a stride of one over there, and then whatever is the size of the input the size of the output, x, y size of the output is going to remain the same. The number of channels will obviously vary down by the number of channels which you put down in your convolutions over here, and that is the simple trick which it was doing. So, at the same kind of a pyramid level of resolution you take on more number of features and eventually keep on stacking them.

(Refer Slide Time: 25:43)



Local Response Normalization (LRN)

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

Now, local response normalization as is goes is something of this sort. So, the whole idea is that, over a window length of n. So, say I have some N number of channels as over here. So, initially on my first input I had a 3 channel input. So, it was 2 2 4 cross 2 2 4 cross 3. the next one it goes down and makes it 64 channels. So, convolution of 3 cross 3 and 64 channels of it.

Now, the output over there of 64 channels, where you have 224 cross, 224 cross 64 is the output of the first, after the first convolution, on any of the variants abcd. Now, there I have these 64 channels present over there, if I look at one particular location over there and say this location is my x, y location and I look into my jth channel. Now I would like to normalize the response of that one in total.

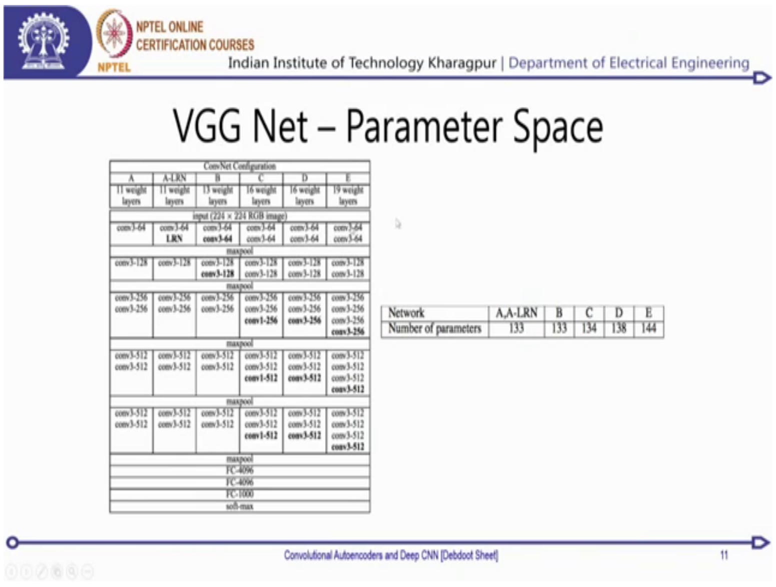
So, what this would mean is that if one of these channels cannot suddenly be abruptly high or abruptly low as compared to its neighbors. So, now, I would have to have some sort of a moving window, which operates on a per pixel basis across the length of the channel and that is this local response normalization which prohibits, any sudden change along the channels at a given pixel location.

So, one is it you are convolving in the x,y space. So, your xy ways of changes are pretty much limited. You will not be getting on noisy patterns, but then I do not even want noisy patterns along my channel dimension and for that I would be putting a local response normalization. This does have a very significant role in case of VGG Net to bring down all responses onto a conformal dynamic range, in which it works out.

So, as we keep on doing the exercises, you will see and we will explain you, how this has an important aspect, and again like when we try to visualize these kernels you would be seeing that why this is important to have a local response normalization, and but then it was eventually figured out that this is a too complicated, computationally expensive process, actually which where you also need to have these factors of alpha and beta, which as of now the authors had just chosen empirically and just put over there, but this k alpha and beta these factors are also something critical which can be learnt.

And, but then their learning is something which again introduces a lot of variability in those system. and eventually down the line people have sort of restricted using local response normalization and have come down with another really interesting concept called as a batch normalization. So, that is not a substitute for local response normalization in any way, but batch normalization is definitely an interesting fact to also go and study.

(Refer Slide Time: 28:24)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

VGG Net – Parameter Space

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 11

So, one point which these authors had put down and which we also make an important aspect to study is, what is the total number of weights or that is what is called as a parameter space.

So, now on when this networks start getting bigger and bigger. So, you will have a major challenge in terms of fitting them within your available CPU and GPU memory. Now if the model parameters and weights they just explode out, then you cannot pretty much train them within your network and if you cannot load your network under your CPU then where are you going to do it.

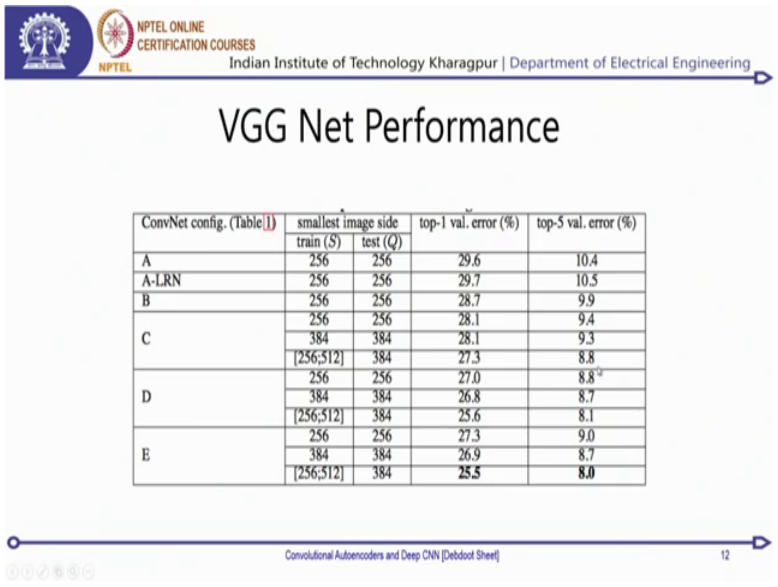
So, we need to really calculate out what is the weights and that will guide me, whether I can use given my hardware of computing, can I use an 11 layer network or can I use a 19 weighted layer network. and the way it’s done, if you remember that if when we were doing down with the fully connected neurons in multi layer perceptron. So, if n number of neurons are connected to m number of neurons and you have one single bias coming over there, then what happens is basically that for each of these. So, each of all the n neurons are connected to each of these m neurons and there is also one bias which goes down over there.

So, there is basically n plus 1 cross m is the number of neurons which are connecting. So, now, here what you need to do is, that if I have convolutional layers. So, this is a 3 cross 3 layer and there are 64 such channels along with each channel that is also one bias. So, it makes it 3 cross 3 plus 1 into 64. So, 3 cross 3 is 9, 9 plus 1 is 10. So, 10 into 64. So, one of this will have 60 40 such bits which are to be learned down. So, the next one over here will have 10 into 128 or 1280 one thousand two hundred and eighty weights to be learned, this will have 2560, this will have 2560 these kind of weights. Also you need to figure out that over here, I had a 3 cross 3, but then my input was a something which was 224 cross 224 cross 3. So, that meant my number of weights was 3 cross 3 cross 3.

So, I have 3 cross 3 in the x direction and y direction and 3 number of channels in my z direction which is my color. So, this volume total, the number of weights over there is 3 into 3 into 3 and that is 27 then a bias which is 28. So, this becomes 28 into 64. Now the output over here is 64 channels. So, over here whatever kernel goes in; that is 3 cross 3 into 64, So, 64 into 9 added to that is a one for your bias and then you have 120 order of those. Now together if you do this total summation, you would be getting down that you have about 133 million parameter, not million I think its 1000, so its 133000 parameters, which you have over here in this model to learn. the next one is 133000 parameters to be learnt in module B.

So, this is a whole detailing is, where it’s given down on the paper. So, I would definitely suggest that you go through and do it. we will also have exercises and a bit of solid examples over there, when we do the codes. So, we will be writing down the network and then there is a even of sort of a code equivalent way of finding out what is the total number of parameters, which we can learn down over there.

(Refer Slide Time: 31:48)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

VGG Net Performance

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

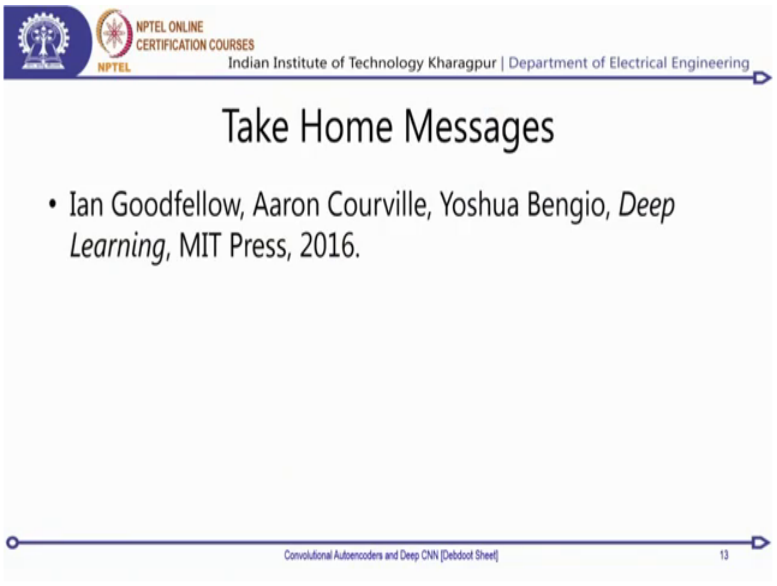
Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 12

.

So, with that like we are almost said, the end of it with just looking into the performance. So, they have been trying to do what is a top 1 percent and a top 5 percent and what is the total amount of error. So, the error as you see, as it goes down with a much deeper network goes really low. So, it is with the top 1 percent or the best, prediction over there is over, 25.5 percent error, and the top 5 percent evaluation of the error goes really down to 8 percent or even lesser than.

So, this was the first like really early days version of it and todays networks much deeper networks, and something we just came out last year the dense net is, which beats everybody in the game. So, we have those in the next week itself.

(Refer Slide Time: 32:31)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Take Home Messages

- Ian Goodfellow, Aaron Courville, Yoshua Bengio, *Deep Learning*, MIT Press, 2016.

Convolutional Autoencoders and Deep CNN [Debdoot Sheet] 13

So, there we get to an end and for more details you can definitely go through the textbook on this particular topic as well as all the papers which have been citing down, and then the next class onwards we do more of coding exercises, and there would be a tutorial at the end of it, where we would be solving for any kind of a given network, then how do we calculate and find out what is the total number of weights and the parameters which we need to do, and how to gauge that number of weights and parameters on to finding out.

What is the compute requirement, which I will have to have, what is the total number of operations or mathematical operations I will have to do in my forward and back propagation as well as total number of memory which, I will be requiring over there, which is critical in terms of understanding your computational limitations given down a particular hardware, that you always do not need to have access to a GPU, may be a simpler network can actually work on a CPU and how to do that.

So, they will be down a bit at the later part of this week when we have it done. So, with that we come to an end with this interesting discussion on some early concepts of convolutionaldeep convolutional neural network. So, stay tuned for the rest of them and then all the best.