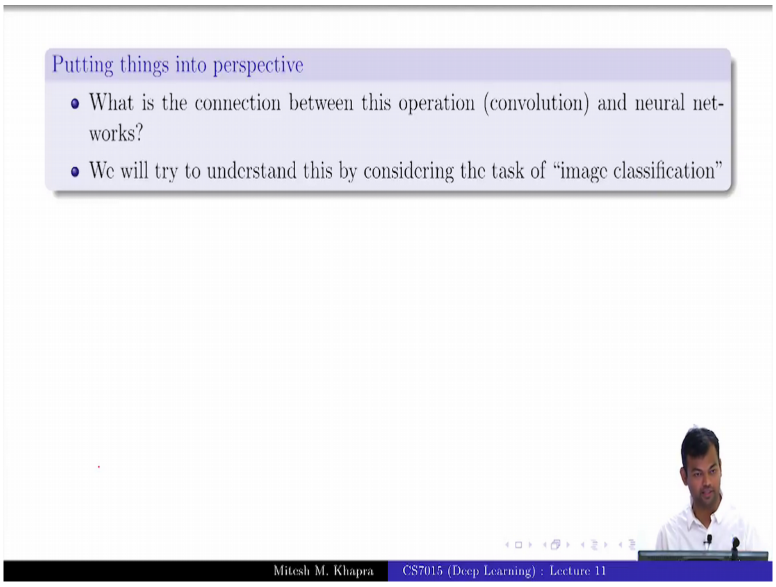
Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 11
Convolutional Neural Networks

Ok. So now, we will go to the next module, this is for the camera and this is on Convolutional Neural Networks.

(Refer Slide Time: 00:17)



Putting things into perspective

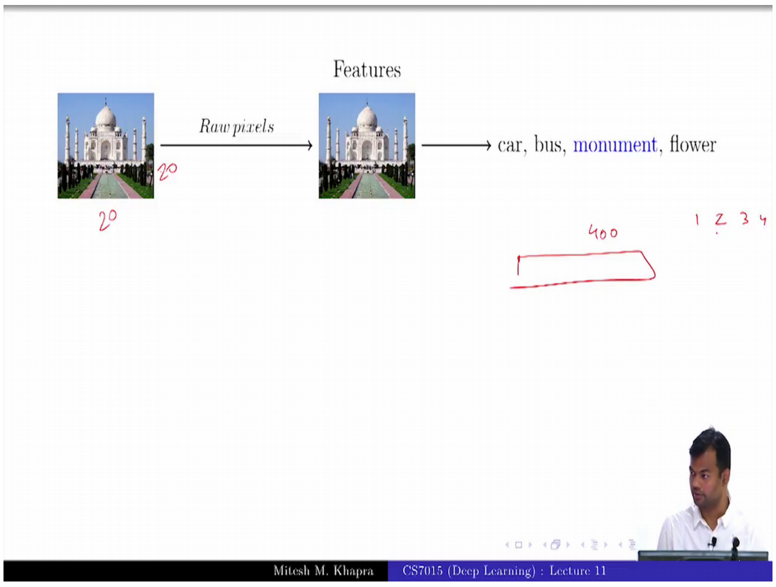
- What is the connection between this operation (convolution) and neural networks?
- We will try to understand this by considering the task of "image classification"

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

So far, we have done what we have just talked about a convolution operation, you just taken some input boxes and converted them to output boxes, what does this anything of this have to do with neural networks? I keep saying that is a course on neural networks right. So, everything has to link to that. So, what is the connection?

So, we will try to understand this by taking the example of image classification and I will use the same trick to get everyone's attention. So, the next few slides are going to tell you the difference between machine learning and deep learning ok. So now, everyone will pay attention.

(Refer Slide Time: 00:49)

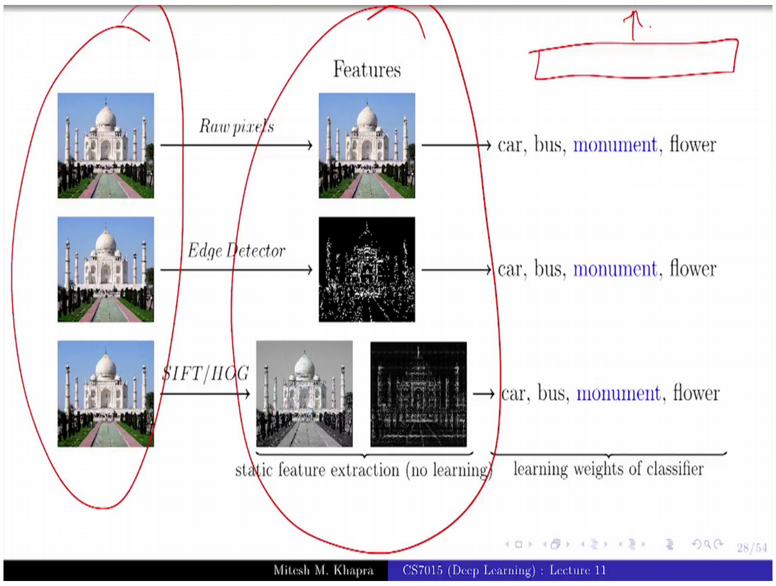

Features
Raw pixels
car, bus, monument, flower
20
20
400
1 2 3 4
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11

So, this is the task you have give given an image and you want to classify it into one of k categories and I am considering 4 categories here car, bus, monument, flower ok. What is the simplest thing that you can do? Suppose this is a 20 cross 20 image you know the simplest thing.

Student: Sir.

Given on the slide; you will just take this as a 400 dimensional input feature vector right and you will treat it as a 4 class classification problem. Train some multiclass SVM or anything on that right. So, you have a simple input so, you are given some 1 million images, all of these are 400 dimensional and they come from 1, 2, 3 or 4, these are the 4 classes, which is car, bus, monument and so on.

(Refer Slide Time: 01:43)


Features
Raw pixels
Edge Detector
SIFT/HOG
car, bus, monument, flower
car, bus, monument, flower
car, bus, monument, flower
static feature extraction (no learning)
learning weights of classifier
28/54
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11

So, you can just treat this as an input feature vector and do your classification right that is the simplest thing that you do or else what you could do is; you could do some kind of feature engineering right, you could say that actually this entire blue sky is not really helping me in deciding anything, these entire green lawns and all this is not helping, if monument, car, bus and flower are the classes, what I care about is the shapes, I do not care about the details inside the shapes, I am not trying to decide whether the car is of a blue color or what model the car is and so on right, all I want to see is that this particular shape of a car is present or not?

Now, what kind of filter gives you the shape of the image?

Student: (Refer Time: 02:16).

Edge detector right so, I could use edge detector. So now, this is something that I have used based on my domain knowledge, that for these 4 classes actually, just detecting the shape is important. So, I will ignore everything else. So, there is a lot of details there right. So, I have actually sparcified my entire input, I have just looking at the edges in the input and now this is a better refined feature as compared to the earlier feature, how many of you agree that this is a more refined feature representation right.

But this was handcrafted, I actually hand coded the edge detector kernel, because I knew that it is 8 at the center and minus 1 everywhere else right, that is how I thought of it that that is what an edge detector is or at least I read about it somewhere right. So, that is how you would do it. So, this is feature engineering.

So, what is this? This is how you do machine learning right, you take an input you do some feature engineering and then you train a classifier on top of that, but now you could become even more creative with the feature engineering and that is what the computer vision community was doing largely before 2012. Come up with different ways of capturing better and better features from images. So, too popular in features from that era and that is I am just talking about 2012 not some like 500 years back, but from that era was SIFT and HOG features, which actually tell you how do the gradients of these pixels change across the image right.

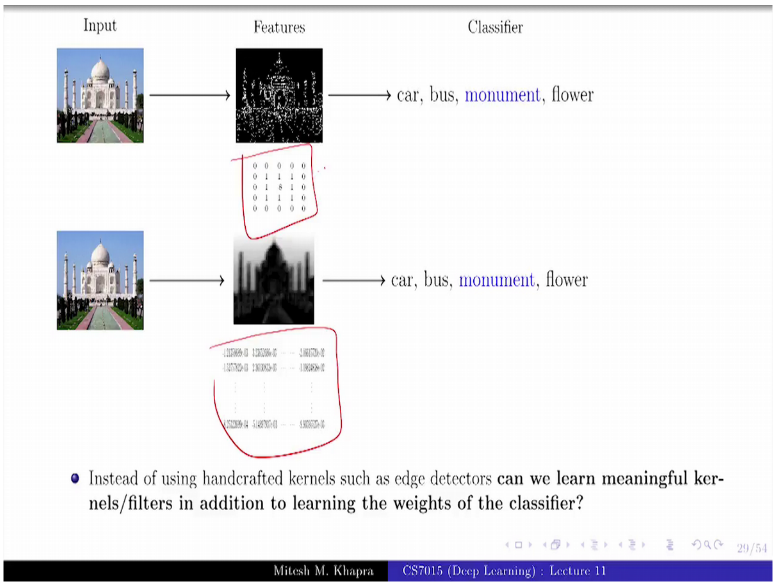
So, this is again try to capture something like how, what is the variation in the image from pixel to pixel right. So, that is the essence that how is you do not care about these entire blue patterns, because they are just telling you sky it is redundant right, if you have seen some 10 pixels or 20 pixels, which has sky you know that a large part of it is going to be sky.

So, these try to capture some abstractions from the image and these are better than the edge detectors and these features were extremely popular. So, what you would do is you take your original input, this is a deterministic algorithm, you apply the HOG algorithm or the SIFT algorithm and it gives you a transformed representation for the image and you can use this transform representation to do classification. And a lot of work prior to 2012, 2011 show that these features work extremely, well across a wide variety of across a wide variety of image tasks ok.

So again, what was happening here? This was feature engineering because someone realized that what I care about is this gradation in the input images and I can capture this by this nice algorithm called SIFT or HOG of course, someone came up with that algorithm, but it is still kind of feature engineering right.

So, this is how the learning is to happen is you are given some input, you do a static feature extraction no learning. So, feature extraction is deterministic you take the input pass it through one of these algorithms; either the edge detector or the blur detector or SIFT or HOG and you get some representations for the input. And the only learning that happens is on top of this transformed input. So, you now have a transformed input and on top of that you are going to train a classifier and you are going to learn the weights of the classifier. So, the only thing that you learn is the weights of the classifier.

(Refer Slide Time: 05:25)


Input

Features

Classifier

car, bus, monument, flower

0 0 0 0 0
0 1 1 1 0
0 1 8 1 0
0 1 1 1 0
0 0 0 0 0

car, bus, monument, flower

- Instead of using handcrafted kernels such as edge detectors **can we learn meaningful kernels/filters in addition to learning the weights of the classifier?**

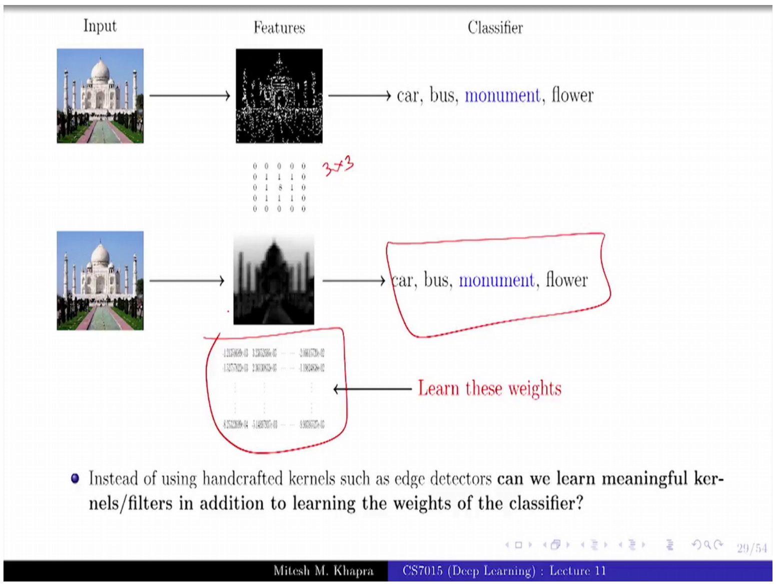
29/54

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

So, that is equivalent to learning only the soft max layer, in case of a feed forward network that is the output layer right

Now, instead of using these so, this is the question instead of using these handcrafted kernels or features, such as edge detectors or blur detectors or what not; can we learn meaningful kernels in addition to learning the weights of the classifier, do you get this? Question at least, whether the answer or not, but you get the question. So, what I am asking is that why should I hand code this edge detector ok.

(Refer Slide Time: 05:55)


Input

Features

Classifier

car, bus, monument, flower

0 0 0 0 0
0 1 1 1 0
0 1 8 1 0
0 1 1 1 0
0 0 0 0 0

3x3

car, bus, monument, flower

Learn these weights

- Instead of using handcrafted kernels such as edge detectors **can we learn meaningful kernels/filters in addition to learning the weights of the classifier?**

29/54

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

Why not have after what is the edge detector; it is like a 3 cross 3 matrix right and I have seen tons of such matrices in my feed forward neural networks, I have dealt with very large matrices, which were called parameters of the network.

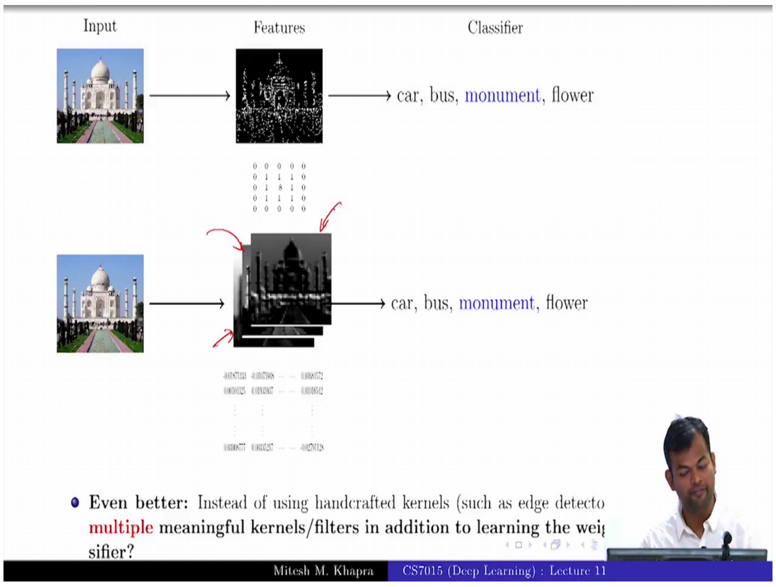
So, why not have a 3 cross 3 or a 5 cross 5 or whatever dimensional matrix and try to learn, what should be the right values in this matrix instead of hand coding the edge detector matrix, do you get the idea, how to do that as still far, but at least do you get the idea that is what I am we are trying to do ok.

So, now instead of just learning the weights of the classifier, we also want to learn the weights of the kernels; that means, instead of using handcrafted features, I am now going to;

Student: (Refer Time: 06:37).

Learn the features. So, that is the difference between deep learning and machine learning. So, you had handcrafted features there and now you are going to learn the representations also by treating them as additional parameters in your network, how you will do that; we will see and it is very simple given that you understand, how to train feed forward networks.

(Refer Slide Time: 07:01)

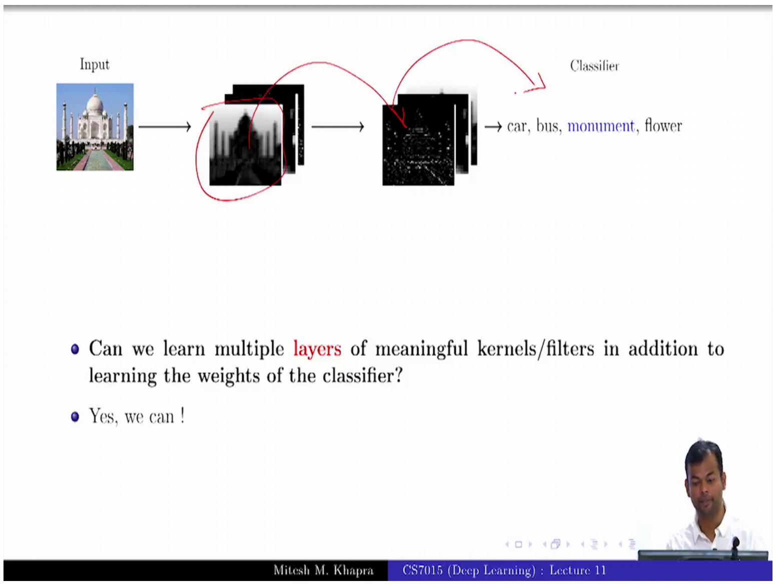


- **Even better:** Instead of using handcrafted kernels (such as edge detecto multiple meaningful kernels/filters in addition to learning the wei sifier?

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

But then why the stop there, why just have one feature representation for the input, can I learn multiple such kernels right. I could have one 3 cross 3 matrix, which learned this first representation another, 3 cross matrix, which learned this another representation and yet another 3 cross 3 or 5 cross 5 or 7 cross 7 matrix, which learns this different representation. So, instead of learning one static representation from the input, I could learn multiple representations from the input.

(Refer Slide Time: 07:29)


Input
Classifier
→ car, bus, monument, flower

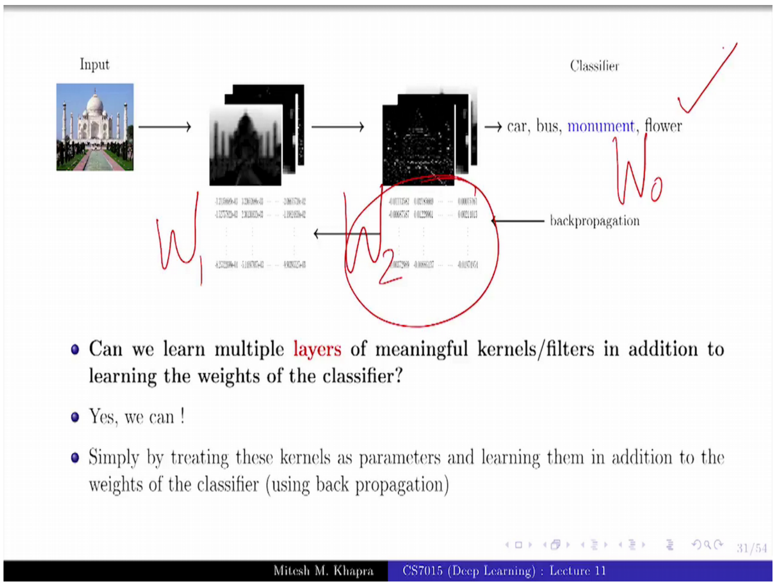
- Can we learn multiple layers of meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11

In fact, why not why just stop there, what is the next thing that I am going to try to do? Multiple layers of features right so; that means, at the first layer I learned this representation, now I could take this and try to learn an even more abstract representation from there and then keep doing this to make it deeper and deeper do you get this ok.

So, at every stage now I have these parameters, which are helping me learn the representation of the input, I am learning multiple representations at every layer and then having multiple layers of these representations right and everything is learnable end to end ok. So, you get the difference between deep learning machine, learning now there is no handcrafting of features, you are learning the feature representation. I know, I understand there is some confusion about how you would do this.

(Refer Slide Time: 08:15)



- Can we learn multiple layers of meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !
- Simply by treating these kernels as parameters and learning them in addition to the weights of the classifier (using back propagation)

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

But we will get to that just trust me on that that you will be able to figure out how to do this ok.

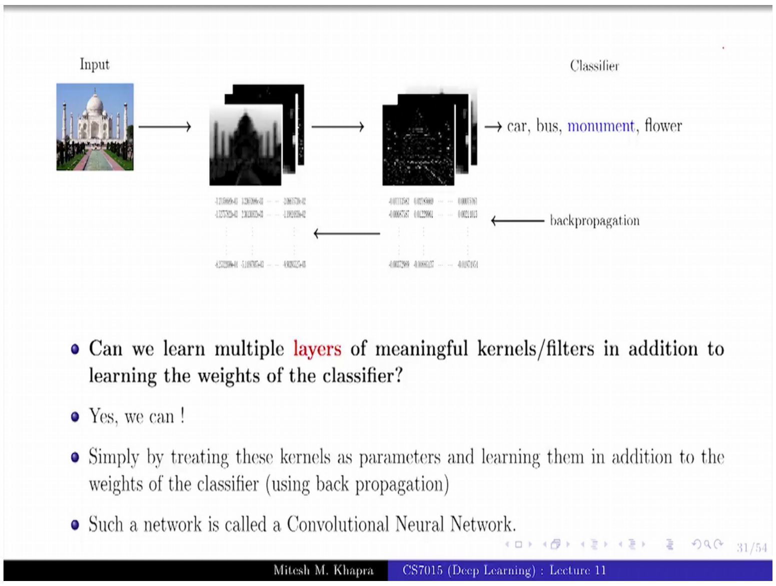
And all of this we have some weight matrices here, we have some weight matrices here, these are the layer 1 weight matrices, the other layer 2 weight matrices and these are the output layer matrices and you see this layer wise, arrangement of these weight matrices and they are very comfortable with this, because we have done feed forward neural networks, where we had these multiple layers and we knew how to back propagate from the last layer to the first layer.

Now, what I am trying to say is that, I would like to adjust these weights of filters in such a way that my classification loss is minimized and what is the loss function that I am going to use here?

Student: (Refer Time: 08:51).

Cross entropy, because this is a multi class classification problem ok.

(Refer Slide Time: 08:59)



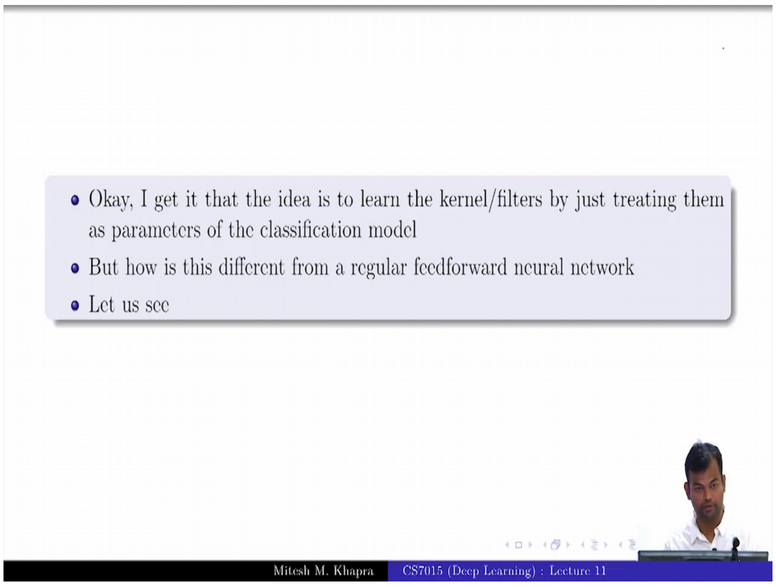
- Can we learn multiple layers of meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !
- Simply by treating these kernels as parameters and learning them in addition to the weights of the classifier (using back propagation)
- Such a network is called a Convolutional Neural Network.

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11 31/54

So, just hang on with this intuition and we will make it more clear fine.

So, such a network which has these multiple convolution, learned convolution operations at every layer and then multiple such layers is known as convolutional neural network.

(Refer Slide Time: 09:09)

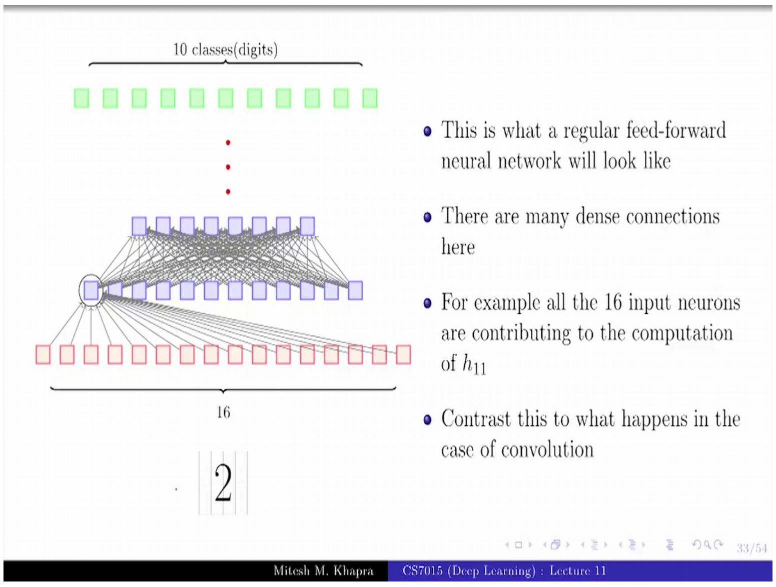


- Okay, I get it that the idea is to learn the kernel/filters by just treating them as parameters of the classification model
- But how is this different from a regular feedforward neural network
- Let us see

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

Ok fine. So, get this idea that we need to learn kernel filters by just treating them as parameters of the classification model ok, but how is this different from a regular feed forward neural network? You could have taken a regular feed forward neural network and I will show it to you on the next slide and what is the difference between that and a convolution operation?

(Refer Slide Time: 09:29)



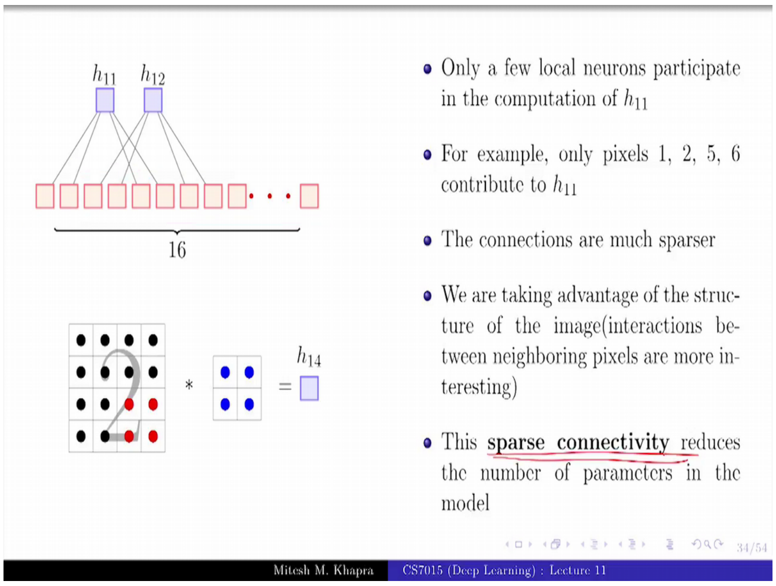
- This is what a regular feed-forward neural network will look like
- There are many dense connections here
- For example all the 16 input neurons are contributing to the computation of h_{11}
- Contrast this to what happens in the case of convolution

So, if you understand that then, you would be done for this lecture.

So, we have an input. So, let us say now, I will take back the eminist case, where you are given an input as an image and these are digit inputs and you want to classify them into 1 of 10 inputs and I am going to assume that, my input is 4 cross 4; that means, I have 16 pixels ok.

So, the simplest thing that, I could have done or the feed forward neural network way of doing this is that, I would just flatten out this image, I will get 16 inputs, I need 10 outputs at the output layer. So, I have an output layer, which will have 1 of these 10 classes and then, I add as many layers that I want in between ok. This is what a feed forward neural network would look like and if I consider any one neuron in the first layer, it takes inputs from all the 16 inputs right, that is how a feed forward neural network is, you have these extremely dense connections, where every output depends on every input at every layer ok.

(Refer Slide Time: 10:31)



h_{11}
h_{12}
16
h_{14}

- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}
- The connections are much sparser
- We are taking advantage of the structure of the image(interactions between neighboring pixels are more interesting)
- This sparse connectivity reduces the number of parameters in the model

34/54

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

Now, so, this is the same story which I have said. Now, let us look at what a convolutional neural network looks like. So, again you have these 16 inputs, I am using a 2 cross 2 convolution ok. Now if I use a 2 cross 2 convolution, if I place it here, then I am using pixels 1, 2, 5 and 6 and computing one value. So, you see the difference between this and a feed forward neural network, in a feed forward neural network h 1 1 would have depended on.

Student: (Refer Time: 11:03).

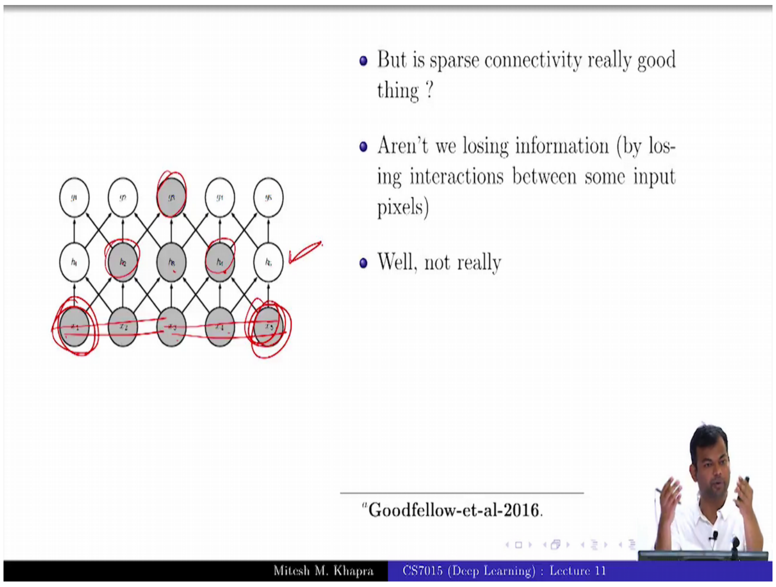
16 values; 16 inputs and a convolutional neural network it is depending on.

Student: (Refer Time: 11:09).

Only 4 neighbors and similarly h 1 2, I am using a stride of 2 by the way right. So, I am not placing the filter here, I am just skipping one step, h 1 2 would depend on pixels 3 4 7 8 and so on right.

So, one thing is clear that as opposed to a feed forward neural network, you have sparser connections here is that clear and why do we have sparse connections? Because we are exploiting our knowledge about images that in an image, you do not really care about the interactions between on between a pixel at the leftmost left top most corner and the right bottom corner right. So, there is sky here, there is ocean here or there are trees here, you would want to capture the neighborhood around that pixel not really capture it with the entire image, that is why you do not want all of these 16 inputs to contribute, you only want a small neighborhood to contribute do you get that intuition ok.

(Refer Slide Time: 12:13)



- But is sparse connectivity really good thing ?
- Aren't we losing information (by losing interactions between some input pixels)
- Well, not really

"Goodfellow-et-al-2016.

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

So, this is the first property of a convolutional neural network that it has sparse connectivity ok, but its sparse connectivity really good? I just made a case for that. Now I am going to counter argue right, is it really good that you have these sparse connections? Because you are losing out information right, you are using out interactions between certain pixels. So, why is that good?

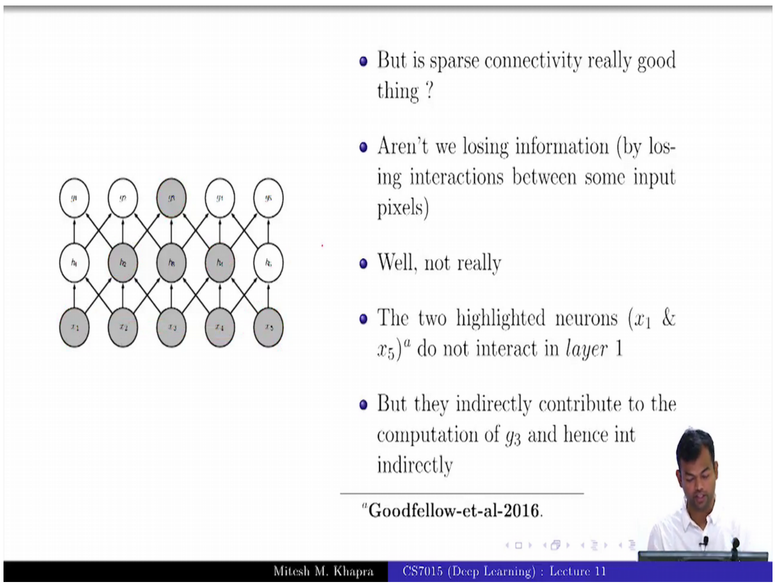
Student: (Refer Time: 12:29).

I am hearing a lot of interesting answers, but remember that you are always going to have multiple layers ok. So, consider these 2 pixels, in the first layer these 2 pixels did not interact, because h 2 only dependent on these 3 and h 4 only dependent on these 3, there is no a there is no unit here which depends on both x 1 and x 5 is that obvious, Because I am just using a window of size 3.

But now once I go to the next layer, once I go to g 3; g 3 depends on h 2, h 3, h 4 here, which in turn depends on x 1 x 2 x 3 x 4 x 5 right. So, even though at this layer x 1 and x 5 are not interacting with each other as you go deeper these interactions become obvious, do you get that right. So, that is why you will always use a deep convolutional neural network, where all the pixels get to interact at a deeper layer, but at the more image, it layers you just want to capture the interactions between a neighborhood.

So, it is like you take this neighborhood find out something, then take neighborhoods of neighborhoods and then try to find out something at the next layer and keep continuing in this layer, how many if you get this? Right ok. So, this is what sparse connectivity looks like.

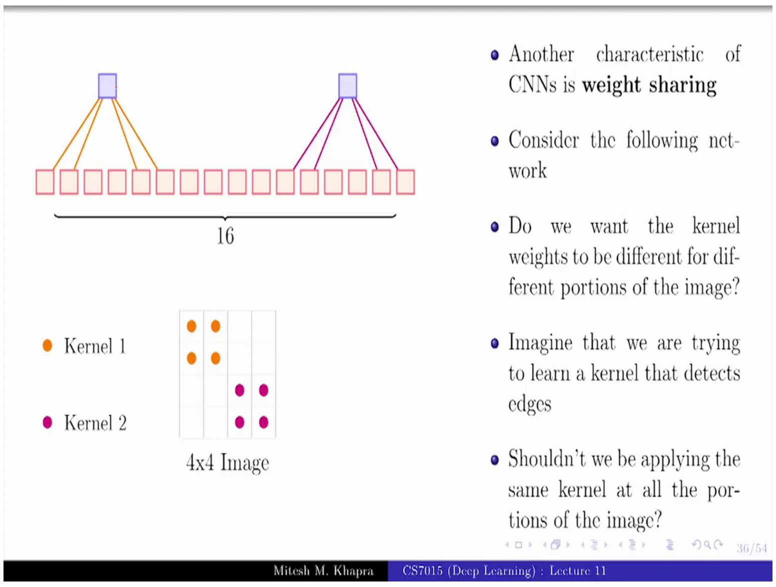
(Refer Slide Time: 13:43)



- But is sparse connectivity really good thing ?
- Aren't we losing information (by losing interactions between some input pixels)
- Well, not really
- The two highlighted neurons (x_1 & x_5)[a] do not interact in *layer* 1
- But they indirectly contribute to the computation of g_3 and hence int indirectly

[a]Goodfellow-et-al-2016.

(Refer Slide Time: 13:47)



- Another characteristic of CNNs is **weight sharing**
- Consider the following network
- Do we want the kernel weights to be different for different portions of the image?
- Imagine that we are trying to learn a kernel that detects edges
- Shouldn't we be applying the same kernel at all the portions of the image?

Another characteristic of CNNs is something known as weight sharing. So, let us see what it is? So, remember I had considered this 2 cross 2 kernel and I was placing it at these 4 pixels, which is pixels 1, 2, 5 and 6 and I was pacing another kernel at these 4 pixels, which is pixels 11, 12, 15 and 16 right, these 4 pixels and I have used different colors for them indicating that these filters are different. So, they are both 2 cross 2 filters, but I am assuming at the values inside them are different. Does this have to be the case? Just think, what a filter is trying to do.

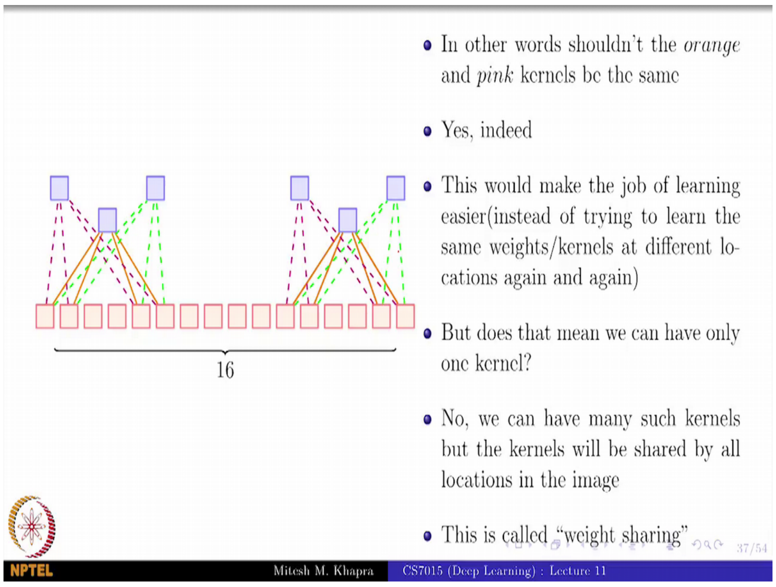
Student: (Refer Time: 14:27).

It is striding across the entire image at every location, I want to do the same operation remember, when we are doing blurring or edge detection or sharpening I had the same filter, which I was applying at every location. So, I want to see, what is the effect of this filter throughout the image? So, I do not really want to change this filter; that means these 4 weights would be the same as.

Student: pink weights.

The pink weights; how many of you get this? So, this is a question, do we want the kernel weights to be different for different portions of the image? So, imagine that we are trying to learn a kernel that detects edges. So, the same kernel configuration is required throughout the image because, that is the kernel configuration, which will detect an edge.

(Refer Slide Time: 15:17)



- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed
- This would make the job of learning easier(instead of trying to learn the same weights/kernels at different locations again and again)
- But does that mean we can have only one kernel?
- No, we can have many such kernels but the kernels will be shared by all locations in the image
- This is called "weight sharing"

So, you want the same kernel to be there everywhere. So, we are going to share these weights, they should not be these pink and orange weights, we will just have the same weights everywhere ok. So, this is known as weight sharing.

So, now this is something ridiculous, if you think about it now how many weights do I have in layer 1?

Student: (Refer Time: 15:33).

4 weights that is all that looks too less right it would lead to.

Student: (Refer Time: 15:39).

Dash fitting.

Student: (Refer Time: 15:42).

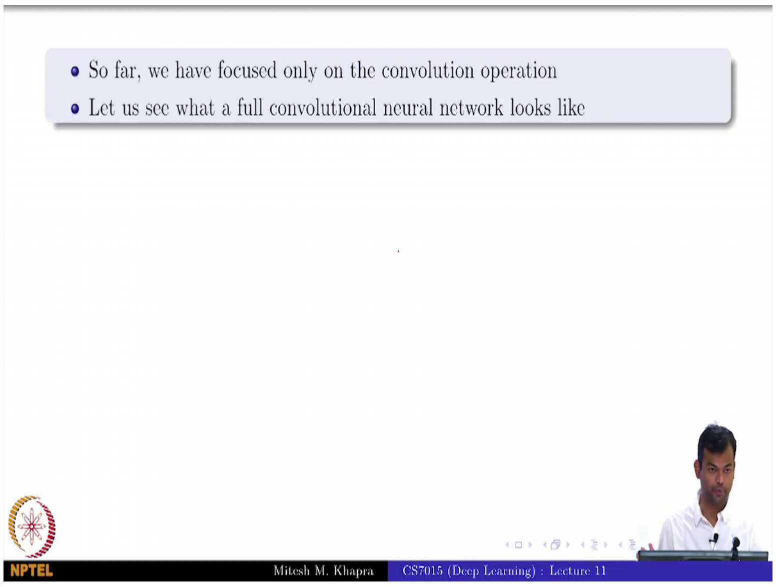
Under fitting because we have very few parameters so, how do I deal with the situation?

Student: (Refer Time: 15:45).

You will have multiple kernels right. So, you will have another kernel, which takes something else you will have one more kernel, which takes something else and you can have as many such kernels right. So, the more the number of kernels will have you will have that many into 4, as the number of parameters and that many outputs at layer 1, how many if you get this? Ok good.

So, these are the 2 important characteristics of convolutional neural networks, one is sparse connections and the other is weight sharing ok.

(Refer Slide Time: 16:17)



- So far, we have focused only on the convolution operation
- Let us see what a full convolutional neural network looks like

NPTEL
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11

So, so far we have focused only on the convolution operation. Now let us see, what does a full convolutional neural network look like or maybe I will do this next time, I think this is.