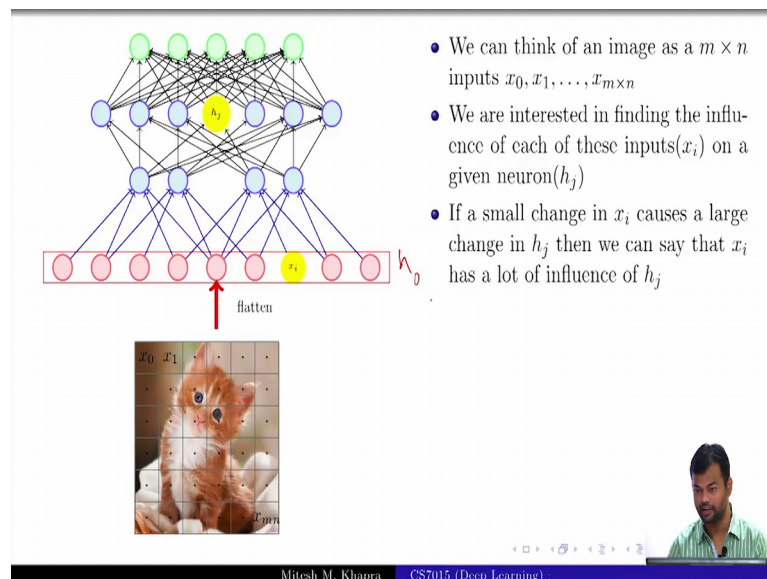


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 96
Finding influence of input pixels using backpropagation

So, now, so far what we have done is, we have seen the influence of neurons on the or what image patches cause a neuron to fire. Then we have visualized the weights. So, neurons have been visualized, weights have been visualized then, we have done some occlusion experiments on the input image. Now, we will take this further and we are interested in seeing that, what pixels in the image actually help in the output or in any neuron in the intermediate layers and we will find out some principal way of finding this influence right and we are going to use back propagation; that means, we are going to use gradients ok.

(Refer Slide Time: 00:50)



- We can think of an image as a $m \times n$ inputs $x_0, x_1, \dots, x_{m \times n}$
- We are interested in finding the influence of each of these inputs (x_i) on a given neuron (h_j)
- If a small change in x_i causes a large change in h_j then we can say that x_i has a lot of influence of h_j

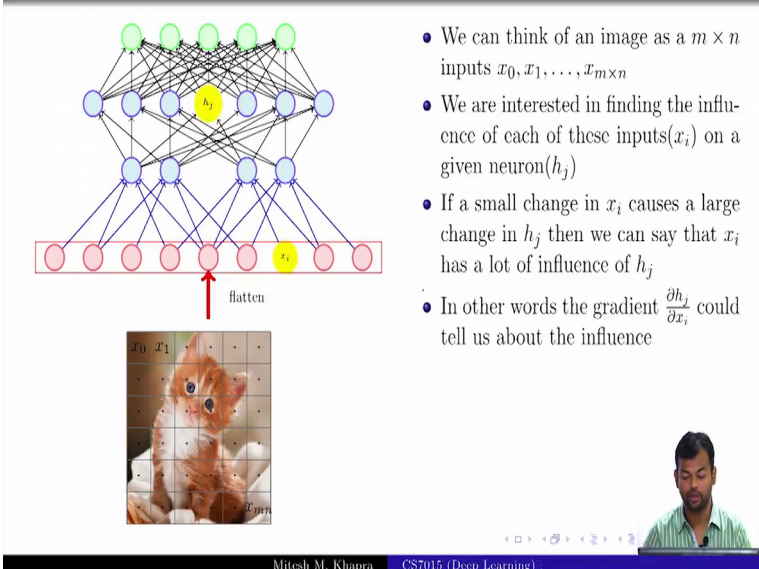
So, we can think of an image as an m cross n inputs, going from $x_0 \times 1$ all the way up to x_m cross n nothing great about that. And we are interested in finding the influence of each of these inputs on a given neuron ok. Now, what is one way of computing influence that you have learned in this course, what is the hero of this course? Gradients, right. So, gradients tell you the influence. So, now, can you tell me if I want to compute what is the influence of this neuron or this input on this neuron, what would you do?

Student: (Refer Time: 01:27).

How do you compute the gradient with respect to the input? We have always stopped at the last hidden layer and the weights before that. So, how will we do that, how will you do this ok? This is a trick question just a hint. Is there a restriction on the chain rule or can you do it? You can just keep adding links to the chain right. So, what is so difficult about that? You already know how to compute the gradients till this point and in fact, you will also know how to compute the gradients till this point..

And what is stopping you from doing it up to this point. What if I just call this h instead of x then, you would not have a problem right. And actually, we call it h right we call it h_0 , we can do it right. It is straightforward, so let us see.

(Refer Slide Time: 02:11)



- We can think of an image as a $m \times n$ inputs $x_0, x_1, \dots, x_{m \times n}$
- We are interested in finding the influence of each of these inputs (x_i) on a given neuron (h_j)
- If a small change in x_i causes a large change in h_j then we can say that x_i has a lot of influence of h_j
- In other words the gradient $\frac{\partial h_j}{\partial x_i}$ could tell us about the influence

Mitesh M. Khapra CS7015 (Deep Learning)

(Refer Slide Time: 02:15)

$\frac{\partial h_j}{\partial x_i} = 0 \rightarrow$ no influence

$\frac{\partial h_j}{\partial x_i} = large \rightarrow$ high influence

$\frac{\partial h_j}{\partial x_i} = small \rightarrow$ low influence

- We could just compute these partial derivatives w.r.t all the inputs
- And then visualize this gradient matrix as an image itself

Mitesh M. Khapra CS7015 (Deep Learning) 13/51

If, I want to compute $\frac{\partial h_j}{\partial x_i}$, I can see that if the $\frac{\partial h_j}{\partial x_i}$ is 0; that means, this pixel has 0 influence on the neuron. If it is large then, it has a high influence if it is small then it has a low influence. So, this is how I will see whether, a pixel has an influence of the certain neurons in the in some of the hidden layers. And this is not restricted to convolutional neural networks as you can see, I am just actually treating it like a feed forward neural network with sparse connections ok.

So, we could just compute these partial derivatives and visualize this gradient itself as an image. So, what do I mean by that is, I am going to compute $\frac{\partial h_j}{\partial x_0}$, $\frac{\partial h_j}{\partial x_1}$ all the way up to $\frac{\partial h_j}{\partial x_{mn}}$ right. So, I am going to compute this $m \times n$ entities and I can just visualize this as an image. Now what do you expect this image to look like? If 0 represents gray colour, what do you expect this image to largely look like, what would you actually hope for? This image should be largely gray because, most of the input pixels should not be influencing a given pixel in the hidden layer right, that pixel should influence by only a small number of pixels.

So, that we can say that, this is the patch which causes it to fire and not that every pixel in the input is causing it to fire because, that is meaningless. So, that does not that is not something that we care about. How many if you get this please raise your hands. So, I will just repeat it, if a pixel fires for every pixel if a neuron in the hidden layer is influenced by all the pixels in the input; that means, it is not really discriminating, it is

not really specialized right. We want neurons which fire only for certain patches in the input, so that we know that this neuron is responsible for this kind of a pattern ok.

So, if I plot this as a image, I would want most of these entries to be close to 0 right because, I want the influence to be 0 ok.

(Refer Slide Time: 04:04)

- But how do we compute these gradients?
- Recall that we can represent CNNs by feedforward neural network
- Then we already know how to compute influences (gradient) using back-propagation
- For example, we know how to back-prop the gradients till the first hidden layer

$$\frac{\partial h_{32}}{\partial x_2} = \sum_{i=1}^3 \frac{\partial h_{32}}{\partial h_{1i}} \frac{\partial h_{1i}}{\partial x_2}$$

$$h_{11} = \sum_{j=1}^4 w_{ji} x_j$$

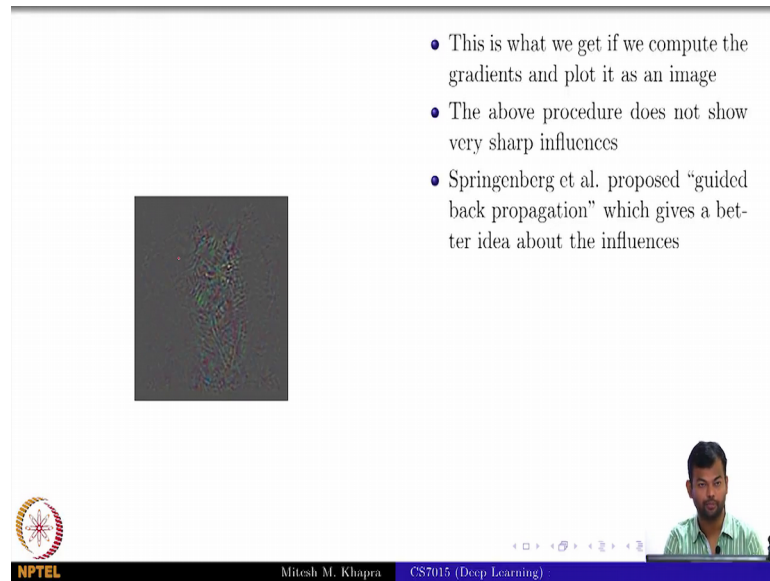
$$\frac{\partial h_{11}}{\partial x_2} = w_{12}$$

Mitesh M. Khapra CS7015 (Deep Learning) 14/51

Now, the question is how do we compute these gradients. So, we will just treat them as a free forward neural network. We already know how to do back propagation across these roots and we just need to add one more term to the chain right. So, I will just show you what we will do here. So, I am interested in $\frac{\partial h_{32}}{\partial x_2}$ or rather from x_2 to h_{32} . So, I will observe that there are 4 paths which go from h_{32} to x_2 or rather from x_2 to h_{32} . So, I will just sum up the gradients along these 4 paths right and if I solve it I will just be left with this ok. So, that is how I will visualize. So, this is very simple we have done a lot of gradients in class, so you can just go back and check this and it should work out well ok.

So, you can just see this and this way we can just compute the gradients for all the input pixels.

(Refer Slide Time: 04:53)



- This is what we get if we compute the gradients and plot it as an image
- The above procedure does not show very sharp influences
- Springenberg et al. proposed “guided back propagation” which gives a better idea about the influences

And now I am going to plot it as a image and this is what my image looks like. Do you see what is happening here? It is all very murky right most of it is great that is fine we expected it. But there is nothing really standing out right even in this patch where you have some non gray pixels, it is almost like the entire cat region is appearing as non gray, the influences are not coming out to be very sharp. We would have wanted something like, only the eye pixels cause some neuron to fire or only the ear pixels cause some neuron to fire and that is not really happening ok. So, it does not produce very sharp influences. So, someone proposed, something known as guided back propagation, which we are going to see next and that helps you to better understand the influence of the input pixels.