# Deep Learning for Computer Vision

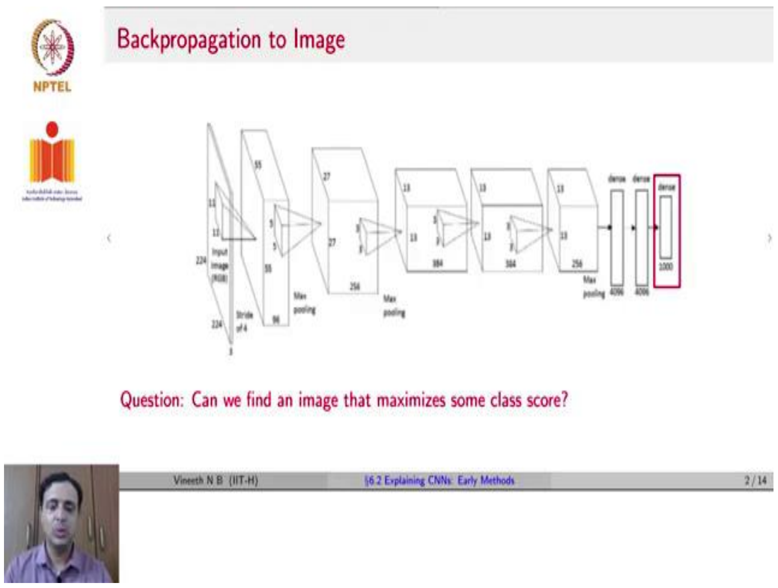
**Professor Vineeth N Balasubramanian**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Hyderabad**

**Lecture 39**

## Explaining CNNs - Early Methods

Continuing from the previous lecture, let us now discuss a few more methods that can help us understand CNN and its predictions.

(Refer Slide Time: 00:27)



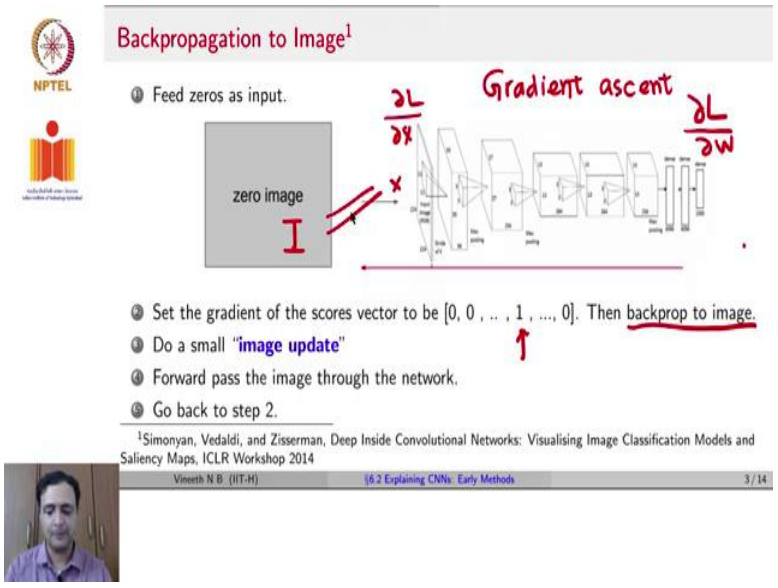
### Backpropagation to Image

Question: Can we find an image that maximizes some class score?

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 2 / 14

Let us start with a question, which is a bit different from what we saw in the previous lecture. The question is, can we find an image that maximizes a class score? So when AlexNet or any other network, when trained on image net has been exposed to a lot of, for example, say cat images, so at the end of the training, does AlexNet know what an average cat looks like? If we ask it to reconstruct an image of a cat, can it do it, is this the question that we are trying to ask? Can you think of how we can do this?

(Refer Slide Time: 01:09)


zero image
I
x
$\frac{\partial L}{\partial x}$
Gradient ascent
$\frac{\partial L}{\partial W}$

## Backpropagation to Image[1]

1. Feed zeros as input.
2. Set the gradient of the scores vector to be [0, 0 , .. , 1 , ..., 0]. Then backprop to image.
3. Do a small "**image update**"
4. Forward pass the image through the network.
5. Go back to step 2.

[1]Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, ICLR Workshop 2014

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 3 / 14

In case you do not have the answer, you can do this using a different use of the gradient descent approach that we used for updating the weights of a neural network. And this was first introduced by a work called deep insight convolutional networks in 2014. And the idea here is you take the trained AlexNet model, and you provide a 0 image as input, a 0 image could be a black image or a gray image, you can choose whichever ones you would like.

And you give this as input to the CNN model, and you now want the final prediction to be one hot vector, where the one is in the position of the cat or any other class that you want to get an image of, and 0s in all other places that is what you would like to see at the last layer, at the output classification layer.

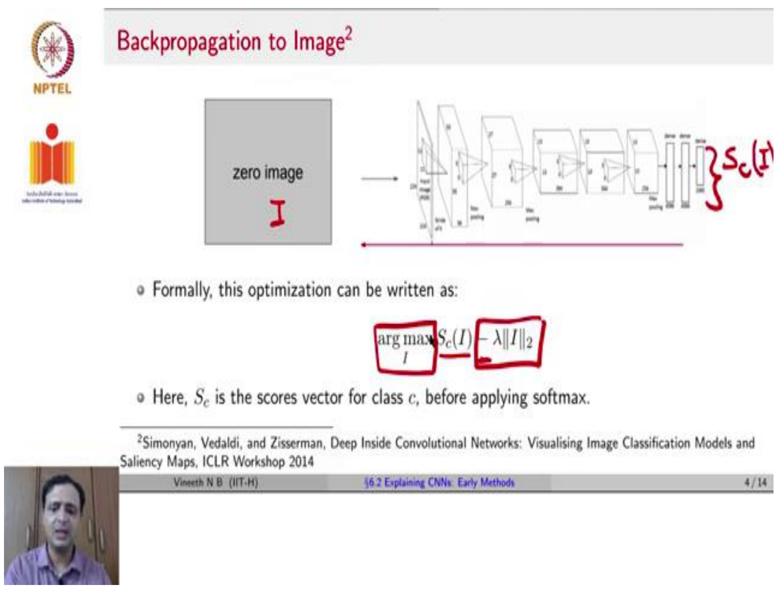
But, when you feed a 0 images input, you will not be able to get that, you probably get a different probability vector. Based on the difference between these two, you have a loss, you can now update your network. But this time, you are not going to update your network, you are going to do what we call back prop to the image. What is a backdrop image?

If you had a loss in the last layer, we have spoken so far, on different ways of computing, $\frac{dL}{dW}$, depending on what weight you are trying to compute the gradient of. If it is the last layer, it is straightforward, if it is an intermediate layer, you use the chain rule, if it is a convolutional layer, you have to think through it a bit differently, batch normalization layer, you have to work these things out, so on and so forth, we have been able to do that.

Can we extend this to compute $\frac{dL}{dX}$, where X is the input? That can be done. This is just another version of the chain rule, you will just have to carefully $\frac{dL}{dX}$ through all the activations and weights in the neural network. If you do not believe it, try it out as homework to work it out. Once you have such a gradient, we try to do an image update, and how do we do the image update? We do it using gradient ascent. So far, we spoke about gradient descent as a methodology to minimize an objective function. In gradient ascent, we maximize an objective function and we will see this objective function in the next slide. So we will use gradient ascent to get the final image.

So once you do a small image update, the updated image. So it could have been an initial 0 image and after doing an update, the x becomes different now, or the I becomes different assuming this image is I which is equal to X for us. Now the I becomes a new iterated update I and that is forward propagated through the network, you again get an output, you compare it to the expected output, which should have 1 at the position of a cat and 0s elsewhere and you repeat this process over and over again.

(Refer Slide Time: 04:32)



## Backpropagation to Image[2]

- Formally, this optimization can be written as:

$$\boxed{\arg\max_{I}} S_c(I) \boxed{- \lambda \|I\|_2}$$

- Here, $S_c$ is the scores vector for class $c$, before applying softmax.

[2]Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, ICLR Workshop 2014

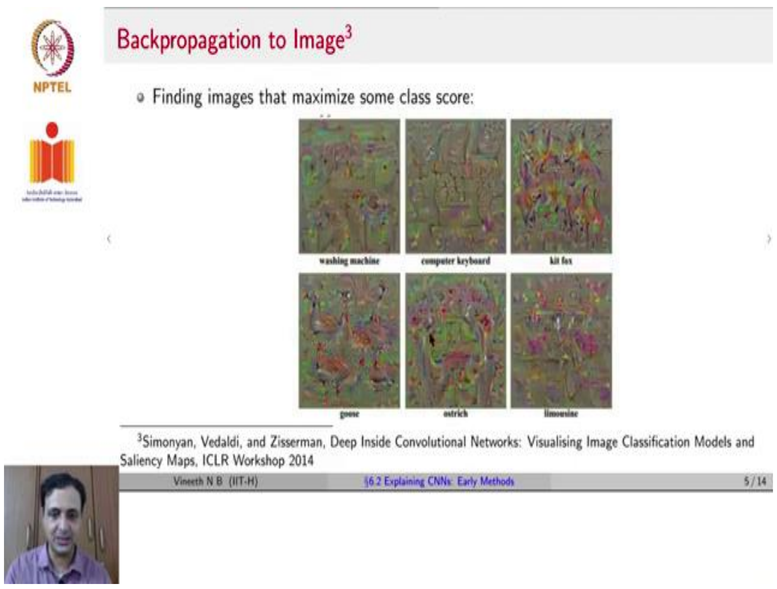
Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 4 / 14

What could be the objective function? The objective function is formally written as you want to arg max over me, where I am the input image, $S_c(I)$, which is the scores that you have in the last layer, $S_c(I)$ is when you propagate I through the network and get scores in the last layer. You want to find an I that maximizes the score corresponding to a particular class.

And you have a regularizer on the image, just so that you will not overfit. Remember, because this is a maximization problem, you have a negative sign because you would ideally like to minimize the two norms of the image that is the reason for the negative side. So how do you solve this problem, because it is a maximization problem, you use gradient ascent.

So you go in the positive direction of the gradient and keep climbing, and keep updating the image over and over again. And at the end of the many iterations, when the gradient between your output, and your expected output in the last layer is close to 0, you would have converged, and you would have got one such image, let us see a few examples.

(Refer Slide Time: 05:45)



## Backpropagation to Image[3]

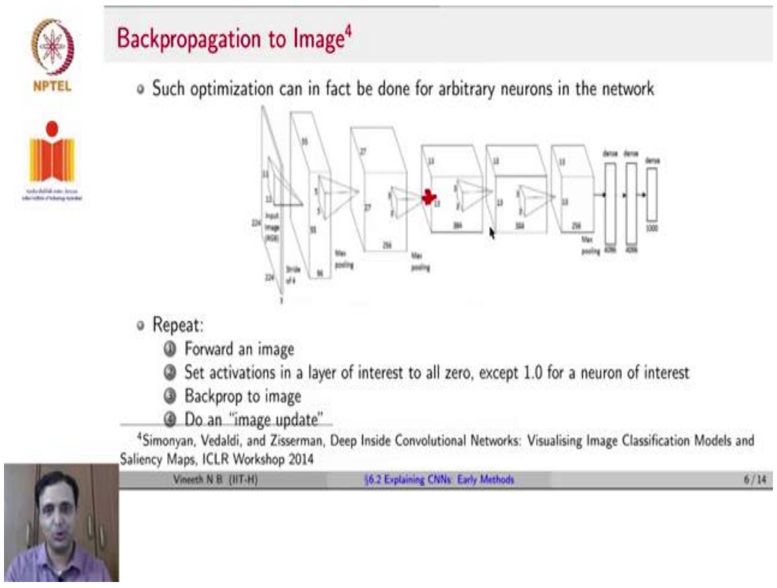
- Finding images that maximize some class score:

[3]Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, ICLR Workshop 2014

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 5 / 14

So here are a few images that maximize a class score. This may not feel like the way we perceive these images. But this is what the model things are representative of those objects. So any such artifacts in an image, it is going to consider it as belonging to an object. So the top left is a washing machine, below is a goose, you can see that there is a structure of a goose located at different points, then you have an ostrich, a limousine, a computer keyboard, and so on.

(Refer Slide Time: 06:21)



## Backpropagation to Image[4]

- Such optimization can in fact be done for arbitrary neurons in the network
- Repeat:
  1. Forward an image
  2. Set activations in a layer of interest to all zero, except 1.0 for a neuron of interest
  3. Backprop to image
  4. Do an "image update"

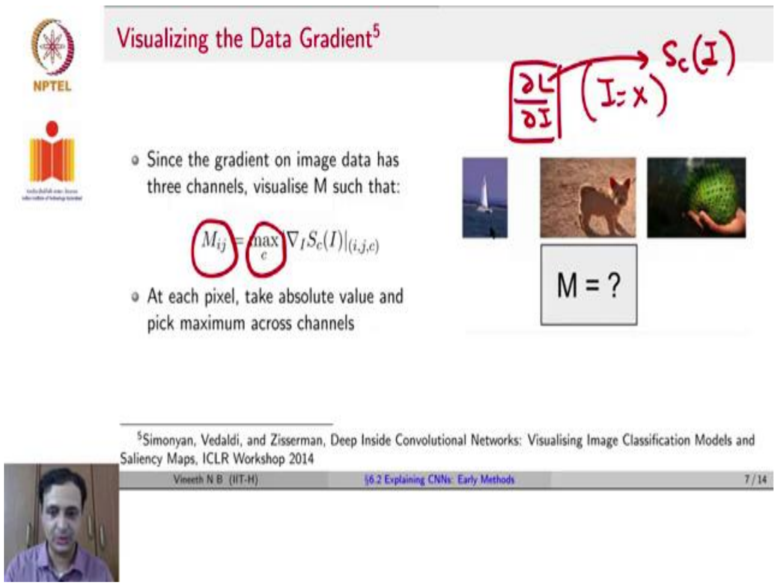
[4]Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, ICLR Workshop 2014

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 6 / 14

You can do such an optimization, not just with respect to the last layer, you can do this with respect to any neuron in the neural network, you can try to see if you want to find out which image is maximally activated when you run, we saw in the previous lecture that we keep forward propagating several images, and then see which all images fire a particular neuron, that would have been one way of doing it.

But now what we can do is start with a gray image or a 0 image, forward propagate your image until a particular layer, let us say you have a particular neuron in one layer that you wanted to maximize, that you wanted to fire for a particular image, then you go forward propagate until that layer, you set the gradient with respect to that neuron to be 1 and everything else to 0, which means you want that to fire and you backpropagate from there and update the input image iteratively using gradient ascent, and you will now be able to get an average image that fires that particular neuron.

(Refer Slide Time: 07:32)



## Visualizing the Data Gradient[5]

- Since the gradient on image data has three channels, visualise M such that:

$$\boxed{M_{ij}} = \boxed{\max_c} |\nabla_I S_c(I)|_{(i,j,c)}$$

- At each pixel, take absolute value and pick maximum across channels

$\boxed{\frac{\partial L}{\partial I}} (I = x) \rightarrow S_c(I)$

M = ?

[5]Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, ICLR Workshop 2014

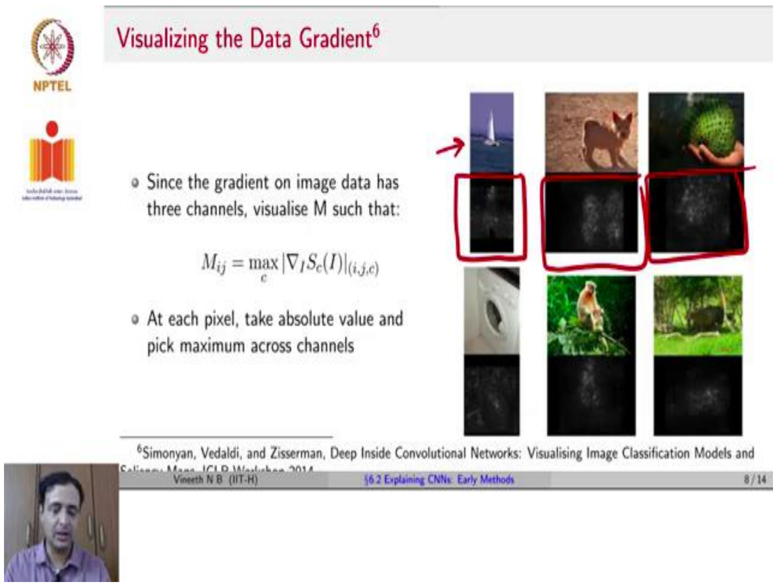
Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 7 / 14

Another added up that was proposed in the same work was the concept of just visualizing the data gradient. So, how do you go about doing it, which means we are talking about $\frac{dL}{dI}$ in this case, remember, we are saying that I is equal to x, which is the input image for us.

So we have a $\frac{dL}{dI}$, which gives us the gradient of the output with respect to the input. So in our case, we are defining l as $S_c(I)$, that is the score that we want to maximize. It is not a loss here, but it is a maximization of the score. But because you have three channels in your input, how do you understand the gradient by itself? You can, this paper suggests that you take an absolute value of the gradient along each channel, and then take the maximum of those as the final gradient at a particular pixel location.

Instead of trying to update an image iteratively, this suggests that simply visualizing that gradient will tell you the shape of that gradient gives you a rough picture of what maximizes that particular output neuron. Because there is color, you take the maximum among those, and you assume that that could be a good representation

(Refer Slide Time: 08:55)



## Visualizing the Data Gradient[6]

- Since the gradient on image data has three channels, visualise M such that:

$$M_{ij} = \max_{c} |\nabla_I S_c(I)|_{(i,j,c)}$$

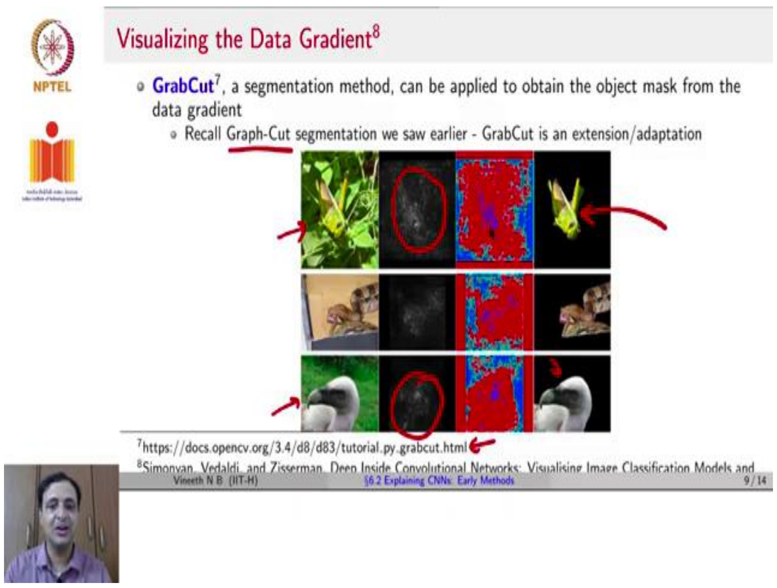
- At each pixel, take absolute value and pick maximum across channels

[6]Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, ICLR Workshop 2014

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 8 / 14

Here are a few examples. So here you see a sailboat and you see that these were the pixels that had the highest gradient across the color channels, which maximize the score. Similarly, this one for the dog and this one for this object, and so on. So what can you do with this gradient? Okay, it does seem to give us an indication of which part of the image is responsible or probably caused the particular probability score to go up.

(Refer Slide Time: 09:30)



## Visualizing the Data Gradient[8]

- GrabCut[7], a segmentation method, can be applied to obtain the object mask from the data gradient
  - Recall Graph-Cut segmentation we saw earlier - GrabCut is an extension/adaptation

[7]https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html

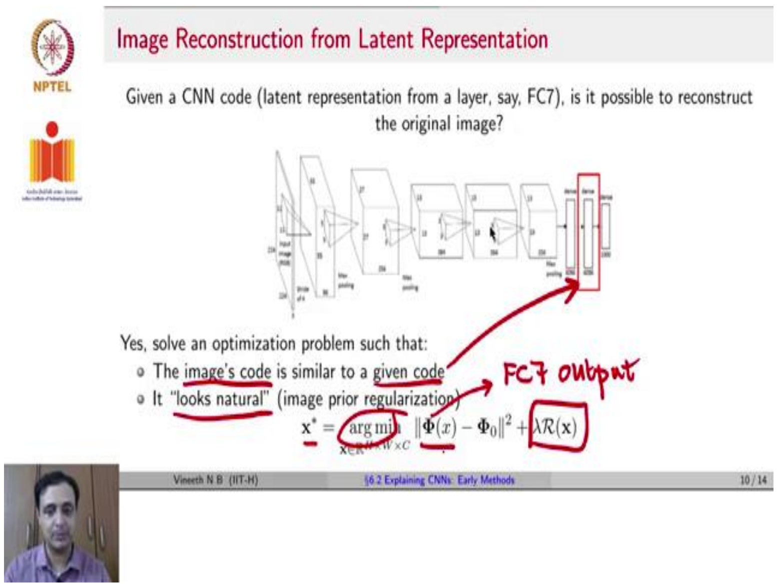
[8]Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 9 / 14

They suggested that you could join this with a segmentation method known as GrabCut, which can be applied to the data gradient to get an object mask. GrabCut is an extension or adaptation of the GrabCut segmentation that we saw earlier in this course. Using this if you had an input image, and you get the gradient corresponding to a particular class, then using those gradients and this GrabCut segmentation algorithm, which is a way of taking those pixels and segmenting the region around them, you end up getting a mask in the input image, which is responsible for a particular class to be predicted. Here are a couple of more examples, so you see especially the third row a bit more clearer.

You can see here that you have a bird, it is a data gradient, then you use GrabCut, to segment out that object from the background, and you get a nice mask of the object, which you can use for other purposes. For details of GrabCut, please see this link below and that is also an exercise for you in this lecture, to see how GrabCut can be used with the data gradient to obtain these kinds of masks.

(Refer Slide Time: 10:56)



## Image Reconstruction from Latent Representation

Given a CNN code (latent representation from a layer, say, FC7), is it possible to reconstruct the original image?

Yes, solve an optimization problem such that:

- The image's code is similar to a given code
- It "looks natural" (image prior regularization)

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^{H\times W\times C}} \|\Phi(x) - \Phi_0\|^2 + \boxed{\lambda\mathcal{R}(\mathbf{x})}$$

FC7 output

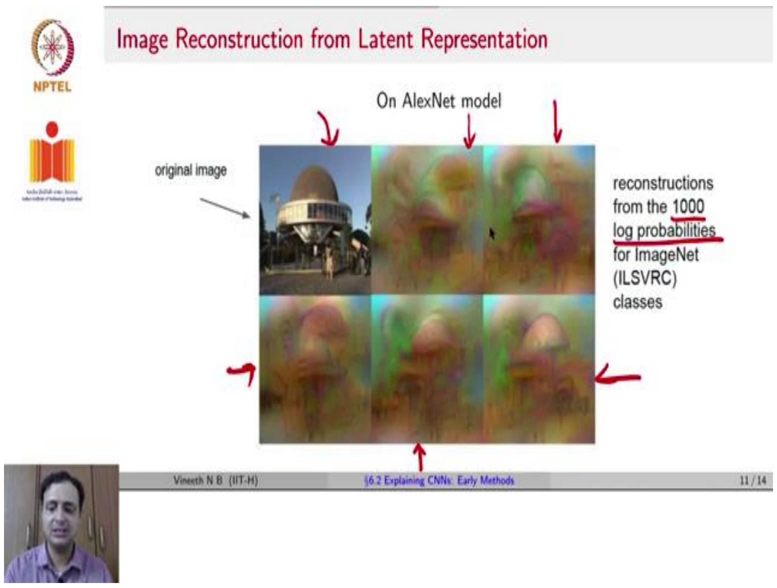
Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 10 / 14

Another question that we can ask you is, given the FC 7 representation of data in a CNN, so that is the output representation of the FC 7 layer or the fully connected 7th layer of AlexNet, is it possible to reconstruct the original image? In the last lecture, we saw that a two-dimensional embedding of these representations from the FC 7 layer does seem to capture semantics and similar images seem to be put together in that embedding space.

But now we are going further and asking if I gave you a code of a particular object, can you reconstruct how that object looks doing some kind of inverse mapping? How do you think you would do this? This can be done again, by solving an optimization problem with two criteria. One, we would like the code of the reconstructed image to be similar to the code that we are given. By code, we mean the representation obtained at the output of the FC 7 layer. And the second is, we want that image to look as natural as possible, we're going to call that image prior regularization or image regularization. So we will keep these two criteria in the objective function, which means what we are going to get is x star is going to be a minimization problem over $\varphi(x)$, take x propagated through AlexNet and take the FC 7 layer output that is what we refer to as $\varphi(x)$. It is a function which we are calling as $\varphi(x)$, we want the $\varphi(x)$ that we optimize through this process to be close to the phi not which is given to us.

Remember, we said there is a code given to us, we want to find an image whose code is close to the code that we have in our hand. So, we are trying to do that using an optimization approach, take an x, $\varphi(x) - \varphi_0$ must be minimized, the mean square error between them must be minimized and you add a regularizer on top of x similar to what we saw on the earlier slide, this is just an image regularization step.

(Refer Slide Time: 13:09)



## Image Reconstruction from Latent Representation

On AlexNet model

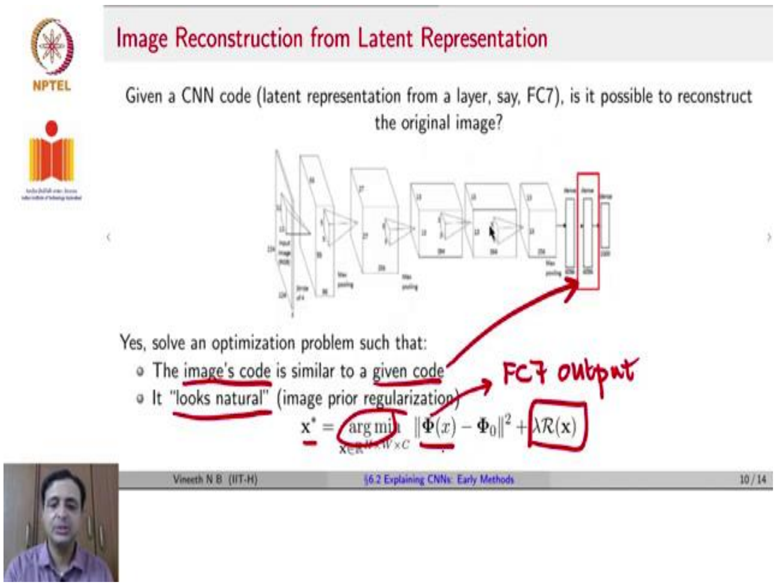
original image

reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 11 / 14

Here are some results using the AlexNet model. So, this is done by taking the log probabilities for the image net classes in the last layer. So, this is the original image, and this corresponds to a particular class in image net, and we are now trying to take that representation for this particular image, and then trying to reconstruct similar images that would give the same FC 7 representations. Why 5 different images?

(Refer Slide Time: 13:45)



## Image Reconstruction from Latent Representation

Given a CNN code (latent representation from a layer, say, FC7), is it possible to reconstruct the original image?

Yes, solve an optimization problem such that:

- The image's code is similar to a given code
- It "looks natural" (image prior regularization)

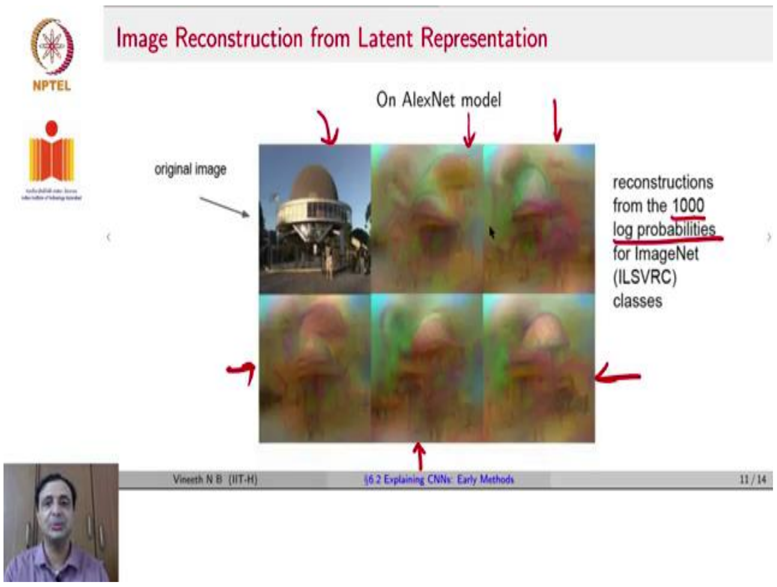
$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^{H\times W\times C}}{\arg\min} \|\Phi(x) - \Phi_0\|^2 + \boxed{\lambda\mathcal{R}(\mathbf{x})}$$

FC7 output

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 10 / 14

If you go back to the previous slide, you see that this is an optimization problem on x, which means you would start with the value of x and then do gradient descent to keep updating x until you got a minimum value on this objective function. So, if you start with different x's, you will get different solutions and those are the different solutions that you see here. So, these are with five different initializations.

(Refer Slide Time: 14:10)



## Image Reconstruction from Latent Representation

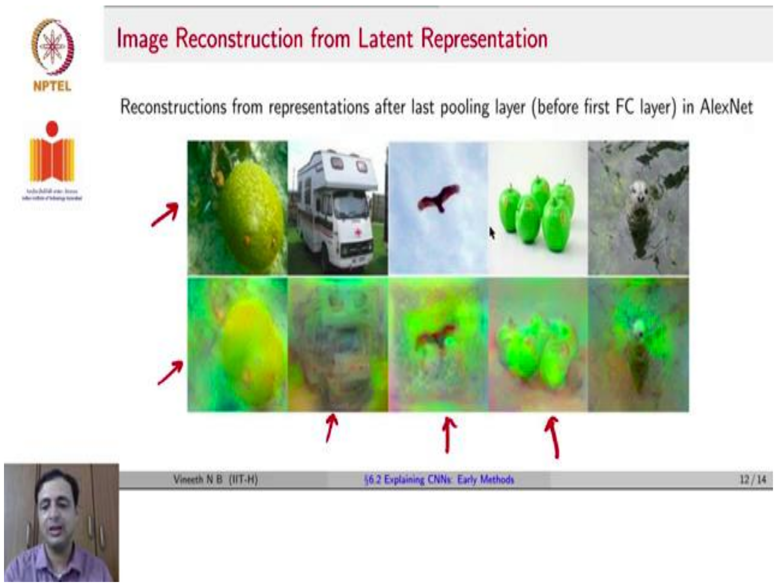
On AlexNet model

reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 11 / 14

And you see that there is an overall similarity to the original image, which we wanted to reconstruct.

(Refer Slide Time: 14:23)



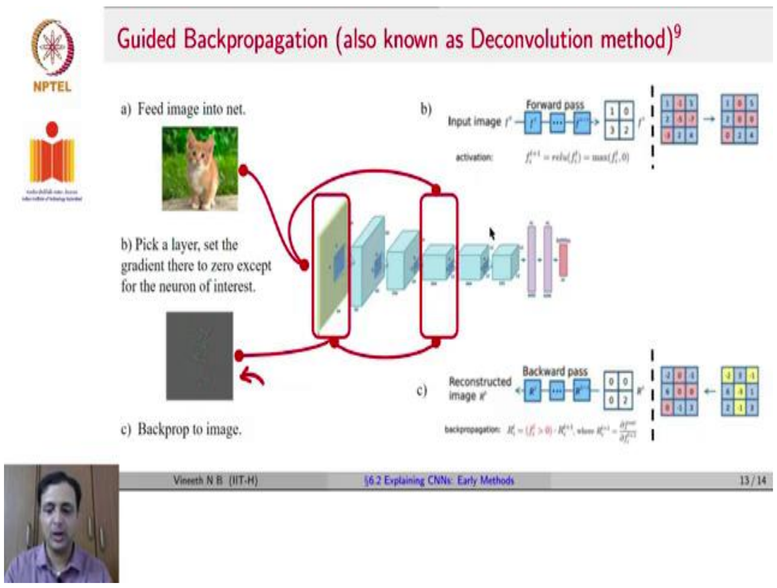
## Image Reconstruction from Latent Representation

Reconstructions from representations after last pooling layer (before first FC layer) in AlexNet

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 12 / 14

Here are more examples, this is the original image, we take the FC 7 representation of that original image, and then ask this kind of an optimization methodology to rediscover that original image, whose representation would have been that FC embedding that was given to us and you see a fairly close reconstruction here. Similarly, this one, this one, this one, and this one.

(Refer Slide Time: 14:53)



## Guided Backpropagation (also known as Deconvolution method)[9]

a) Feed image into net.

b) Pick a layer, set the gradient there to zero except for the neuron of interest.

c) Backprop to image.

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 13 / 14

Moving further, we talk about an important method called guided backpropagation that was developed in 2015 that helped improve the performance while visualizing data gradients. Let us look at this approach. It is sometimes also called the deconvolution method of visualizing and understanding CNNs. But the more popular name today is guided backpropagation. So guided backpropagation is used, along with other explanation methods in more recent years, which we will see in later slides this week. Let us once again start with AlexNet let us and of course, needless to say, AlexNet can be replaced with any other model of CNN, we are only explaining all these methods using AlexNet.

Let us take the AlexNet model. Let us feed an image into the trained AlexNet model. Let us pick a layer and set the gradient there to 0, except for one particular neuron, which we want to maximally activate. Let us assume that this is our setting, so the way we would go about it is you take the input image, you will forward propagate it through as many layers.

And remember, you have a ReLU activation function in AlexNet. And what does the ReLU activation function do? In any layer, where you have such a matrix such a set of activations in any layer, wherever there are negative values denoted by these red boxes here, you replace them with 0 and anything that is nonnegative is retained as it is that's the standard ReLU operation.

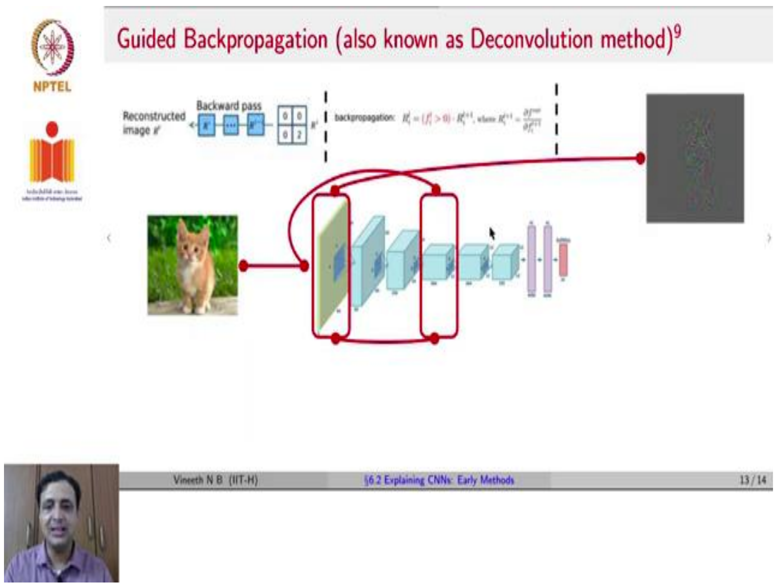
Now, when you do this, you can then backdrop to the image as we just mentioned, because we want to set the gradient of a particular neuron to be 1, everything else to 0 in that layer, and then backprop from there to the image and do gradient ascent on the image to understand which image maximally activated that neuron in that country layer, for instance.

And when we backpropagate, remember that if these were your gradients, right? Let us assume that what you see here were your gradients, we know that, because of the effect of ReLU, wherever there were negative values, these were the locations that had once there, there the gradient would become 0 because when you backpropagate, it will get multiplied by the activations in those locations when you go through chain rule.

And because the activations are those locations become 0, the gradient will also become 0. And this is what you will be left with which you backpropagate further to reconstruct the original image. Note here, that in this particular location, in this particular representation of the gradients, the gradient values can be negative or positive, it is just that where the input was negative, the gradient becomes 0 there.

In the other locations, the gradient can be negative or positive. And when you do this, you can visualize your data gradient and you can see an image that looks somewhat like a cat here, if you observe closely, you can get the gradients that correspond to an outline of a cat. While it has some resemblance to the cat, you can also make out the risk fairly noisy. So what can you do about it?

(Refer Slide Time: 18:19)



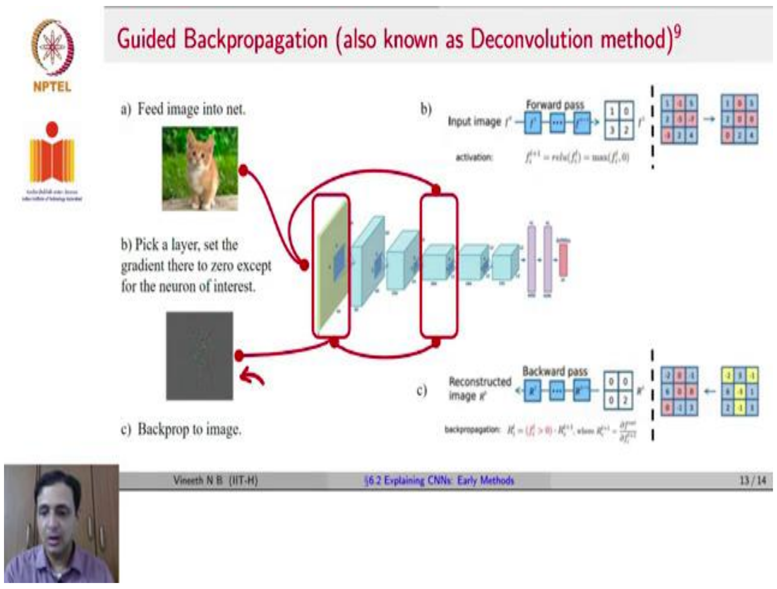
## Guided Backpropagation (also known as Deconvolution method)[9]

Reconstructed image $R^l$
Backward pass
backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$
NPTEL

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 13 / 14

So to handle this scenario, guided backpropagation, which is a method proposed in 2015, proposed that, instead of using in the backward pass, instead of allowing all the gradients to pass through, let us not allow the negative gradients to pass through. What does it mean? We are saying here, originally we said when the input is greater than 0, only those gradients will pass through because the rest of the gradients would have been cut off by the ReLU, which was applied in the forward pass. But now, we are adding that the gradient must also be nonnegative when you propagate.

(Refer Slide Time: 19:03)



## Guided Backpropagation (also known as Deconvolution method)[9]

a) Feed image into net.

b) Pick a layer, set the gradient there to zero except for the neuron of interest.

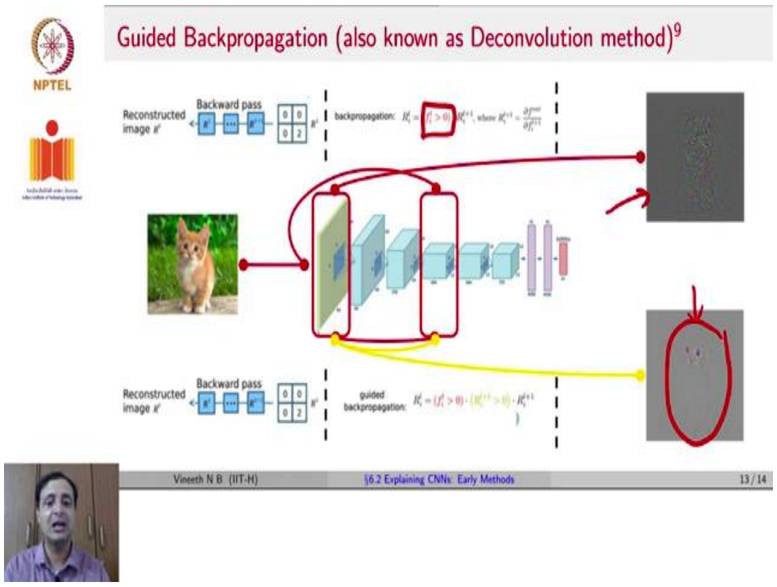
c) Backprop to image.

b)
Input image $f^l$
Forward pass
activation: $f_i^{l+1} = relu(f_i^l) = \max(f_i^l, 0)$
c)
Reconstructed image $R^l$
Backward pass
backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$
NPTEL

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 13 / 14

Rather, if you went to the previous slide. In addition to making these four 0s, you would also make this minus 2 as 0, this minus 1 as 0, and minus 1 as 0, only the positive gradients will be passed through to the previous layers when you do your gradient update for the reconstructed image. And doing this greatly improves the final visualization of the data gradient.

(Refer Slide Time: 19:36)



## Guided Backpropagation (also known as Deconvolution method)[9]

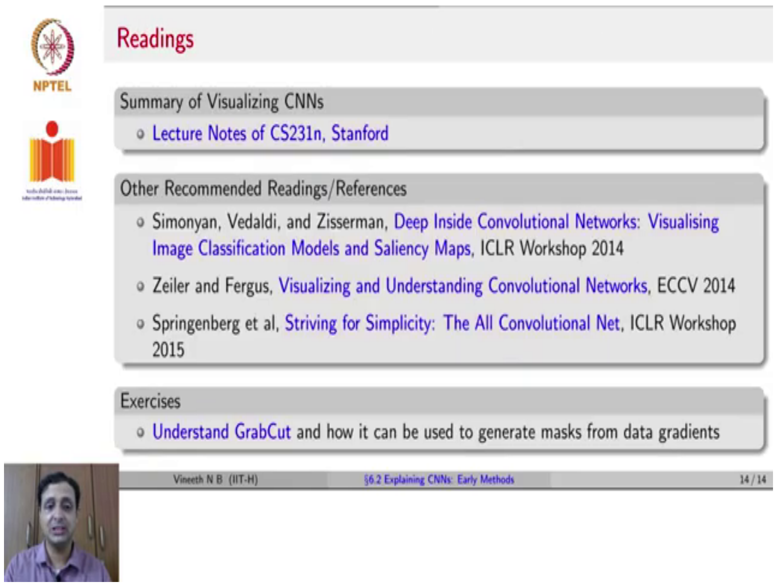
Reconstructed image $x^r$
Backward pass
backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$
guided backpropagation: $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 13 / 14

And now you get a clearer image of the cat the data gradient. Why does this happen? Because you allowed negative gradients to propagate, even aspects of the image that negatively correlated with the outcome in a particular neuron also contributed to coming up with this reconstruction of the image.

By removing those, we now retain only those pixels that had a positive impact on the activation of a neuron in one of the layers that we are interested in. This is known as guided backpropagation and this is something that we use in other methods in the rest of this week's lectures also.

(Refer Slide Time: 20:22)



## Readings

Summary of Visualizing CNNs

- Lecture Notes of CS231n, Stanford

Other Recommended Readings/References

- Simonyan, Vedaldi, and Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, ICLR Workshop 2014
- Zeiler and Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
- Springenberg et al, Striving for Simplicity: The All Convolutional Net, ICLR Workshop 2015

Exercises

- Understand GrabCut and how it can be used to generate masks from data gradients

Vineeth N B (IIT-H) §6.2 Explaining CNNs: Early Methods 14 / 14

The recommended readings for this lecture are once again the lecture notes of CS 231n on visualizing CNNs as well as three papers; Deep inside convolution neural networks, Visualizing and understanding convolutional networks in ECCV, and Striving for simplicity, the all convolutional net, which was the paper that introduced guided backpropagation. And one exercise, use this link hyperlink here to understand GrabCut, and how it can be used to generate masks using data gradients.