



Deep Learning for Computer Vision
Professor Vineeth N. Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Going Beyond Explaining CNNs

(Refer Slide Time: 00:14)





Deep Learning for Computer Vision

Going Beyond Explaining CNNs

Vineeth N Balasubramanian



Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad



Vineeth N B. (IIT-H) §6.5 Beyond Explaining CNNs 1 / 11

For the last lecture of this week, we are going to try to go beyond explaining CNNs. And look at a couple of interesting applications of using the concepts that we saw so far Gradients and various other artifacts for explaining CNNs, but use them in interesting ways. Before we go there.

(Refer Slide Time: 00:41)



Review: Axioms of Attribution¹


Completeness
For any input x , the sum of the feature attributions equals $F(x) = \sum_i A_i^F(x)$

Sensitivity
If x has only one non-zero feature and $F(x) \neq 0$, then the attribution to that feature should be non-zero

Implementation Invariance
When two neural networks compute the same mathematical function $F(x)$, regardless of how differently they are implemented, the attributions to all features should always be identical.

Symmetry-Preserving
For any input x where the two values of two symmetric features are the same, their attributions should be identical as well.

¹Sundararajan et al, Axiomatic Attribution for Deep Networks, ICML 2017



Vineeth N B. (IIT-H) [§6.5 Beyond Explaining CNNs](#) 2 / 11

Let us review one of the home works that we left behind last lecture, which was for you to read axioms of attribution. I hope you had a chance to go through that work. So here are a few of the axioms that have been claimed in the paper which all attribution methods or CNNs methods or visual explanation methods, they are all the same. In this particular context. attribution methods generally refer to methods that broadly cover entire machine learning, maybe even beyond vision, but in the context of computer vision, attribution methods, saliency map, visual explanations, they are all broadly synonymous.

But we would like these attribution methods to satisfy a few of these axioms. The completeness axiom says that for any input x , the sum of the feature attributions equals $F(x)$, which means if you had say N different attributes or 10 different attributes in your input, the sum of their attributions should be the, the actual output that you get of the neural network, the sum of the attributions respect to your neural network should be the actual output.

This is a bit similar to what you see in Deep lift, where we said that the contributions must add up to δt , which was the difference of the output with respect to the reference output. So the completeness property is something that Deep lift satisfies. Another axiom sensitivity states, if x has only one nonzero feature, and $F(x)$ affects the output of the neural network is nonzero, then the attribution to that feature should be nonzero.

And you would actually find that gradient based methods do not satisfy this sensitivity axiom. Think about it and try to answer why you can also read this paper for the answer. When I say Integrated gradient actually satisfies Vanilla gradient based methods do not satisfy the sensitivity axiom. Another axiom is Implementation Invariance. When you have two neural networks that compute the same mathematical function, $F(x)$.


Regardless of how differently they are implemented, the attributions to all features should always be identical. So let us assume that you had two neural networks, one three layer on one two layer, but it happens at the end, that they exactly give the same output for every input that you can give, by some means they have learned the same function, maybe use Relu in one three layer network and use a Sigmoid in another two layer network.


But let us assume they learned exactly the same function. If that happens, the attributions to all features that you get in both of these networks should be the same. And you will actually see that deep lift violates implementation invariance. Once again, the reason, please look at this paper at the bottom. Another axiom states about the symmetry preservation of the attribution method, which states that for any input x , where the two values of two symmetric features are the same, their attributions should be identical as well.


By symmetric features, we mean features that can be swapped and the neural network output will still remain the same. So, this could be in an input this could be an intermediate layer. Generally, symmetry is a broader concept. But we are saying now that if two inputs could have been replaced, and output still stays the same. In that scenario, both those features should have the same attribution.

Please do read this work below for more details about the Axioms. With that, let us go on to these interesting applications that go beyond explanations, but use the concepts that we discussed over the course of this week.

(Refer Slide Time: 04:52)

 **DeepDream²**






- Modifies a given image in a way that **boosts** all activations at any layer, creating a feedback loop
- Any slightly detected dog face will be made more and more dog-like over time

²Mordvintsev et al, Deepdream - a code example for visualizing neural networks, 2015

Vineeth N B. (IIT-H) [§6.5 Beyond Explaining CNNs](#) 3 / 11



The first one is a work known as Deep Dream, which was developed in 2015. Which uses the ideas that we talked about in the earlier lectures of the week. But in a different and interesting way. It modifies a given image in a way in which that boosts all activations at any layer, creating a feedback loop. And which means if you have a dog, a feature or a Kernel that detects a dog's face in a particular layer, you back propagate that on to this given sky image, and keep adding the dog's face to this cloud image. And you will end up getting a pretty interesting artistic image to, to end at the end of this process.

(Refer Slide Time: 05:41)

DeepDream

- 1) Choose a layer/filter.
- 2) Compute the activations for your image up to that layer.
- 3) Backpropagate the activations of your filter back to the input image.
- 4) Multiply the gradients (∇) with your learning rate (α) and add them to your input image
- 5) Go back to 2.

Vineeth N B. (IIT-H) §6.5 Beyond Explaining CNNs 4 / 11

Let us study this method in more detail. So, you have an input image, let us choose a particular layer and a particular filter in that layer. Let us say that this is that layer, and there is a filter in that particular layer. Now you input the image through the CNN, and you forward propagate until that particular layer in the neural network. Once you do that, you back propagate, the activations of that filter back to the input image.

So, you do a backprop to image very similar to what we did in one of the earlier lectures. But this time, the input is not a black image, or a gray image. This time, the input is just any other image that you want to juxtapose or overlay that input onto. So you take the an image of the sky, take a face of the dog or any other filter, and keep adding to this image, the gradient that you get from the face of the dog, you try to see what in the input image would have resulted in that filter being fired, which would be given by the gradient of that filter the respect to the input.

Remember, we talked about, if you wanted to maximize a filter's activation, you said the gradient of only those filters to one, everything else to 0, and backprop to image and update the image using Gradient Ascent, you do the same thing here. And now you multiply those gradients with your learning rate and add them to the input image. That is the gradient ascent that we talked about.

What happens, you now see that you will get an image such as this, in this case, it is not a dog, it looks like a bunch of buildings that you add to the original image. And then you go back to 2

again, you input this image into the neural network, again, maybe you want to use the same filter, maybe you want to use a different filter.

(Refer Slide Time: 07:44)

DeepDream

- 1) Choose a layer/filter.
- 2) Compute the activations for your image up to that layer.
- 3) Backpropagate the activations of your filter back to the input image.
- 4) Multiply the gradients (∇) with your learning rate (α) and add them to your input image
- 5) Go back to 2.

Vineeth N B. (IIT-H) 86.5 Beyond Explaining CNNs 4 / 11

And you can keep playing with this to create different kinds of images.

(Refer Slide Time: 07:47)

DeepDream

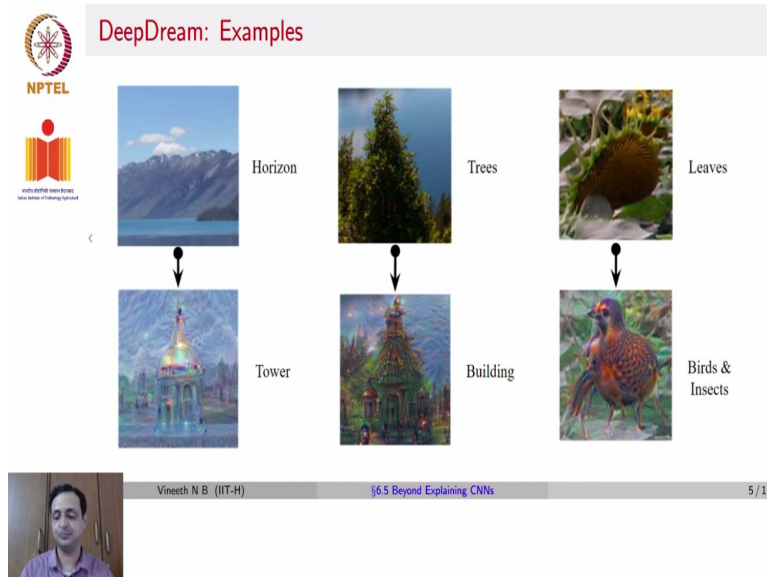
Higher layers produce complex features, while lower ones enhance edges and textures.

Vineeth N B. (IIT-H) 86.5 Beyond Explaining CNNs 4 / 11

You could now take a different filter in a different layer. And now you could add that, and you know you end up getting a different kind of a construction of the image. So, you would see that higher layers produce more complex features, while lower ones enhance the edges and the textures. You can even see that example. In these two images here. This was the top red one

created from a slightly earlier layer. And the yellow one here was constructed by the yellow bordered one here was constructed by a slightly later layer.

(Refer Slide Time: 08:22)



A few more examples of deep dream. So, given an image of an horizon, you can actually convert it to one of a tower, you can actually see that this is the same image taken. But you keep on juxtaposing a tower filter on it. And you end up getting a construction set such as this. Another example is going from trees to keep adding buildings to it. And you come up with a nice pagoda like structure on the same scene. And here is a photo of leaves, and you keep adding the gradients of bird and insect filters. And you get an interesting variant of an image starting with the original leaf image.

(Refer Slide Time: 09:04)


 **DeepDream: Examples**




Admiral-Dog Pig-Snail Camel-Bird Dog-Fish


Credit: Fei-Fei Li and Andrej Karpathy, CS231n course, Stanford, Winter 2016




 Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 6 / 11

More examples.


(Refer Slide Time: 09:10)

 **Neural Style**



 +  = 

Overlay

 Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 7 / 11

And now let us see the other application. Another similar interesting application, which is known as Neural Style, or Neural Style Transfer, as it is popularly known. The idea here is given an image and given a particular style, let us say a modern art style. You want to know how I get this image in this style? For example, can I take my photograph and make it look like Van Gogh had painted it? How would I get that style into my photo? That is what we want to study here. One is

you could simply overlay one of Van Gogh's paintings, overlay that on top of an image that I have, but unfortunately, this really does not give good results.

(Refer Slide Time: 09:56)

Neural Style

Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 7 / 11

Neural Style

³Gatys et al, A Neural Algorithm of Artistic Style, 2015
Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 8 / 11

Another option is to ask a human to do it for us. But that is not what we want to go for. So the third option, what we are going to look for is what is known as Neural Style Transfer, which was also a method that was proposed in 2015.

(Refer Slide Time: 10:12)

Neural Style⁴

Input

Style

1) Extract **input targets** : ConvNet activations of all layers for the given input image.

2) Extract **style targets** : Gram matrix of ConvNet activations of all layers for the given style image.

⁴Gatys et al, A Neural Algorithm of Artistic Style, 2015

Vineeth N B (IIT-H) 6.5 Beyond Explaining CNNs 9 / 11

And let us see how this works. So once again, you have a trained Alex Net model. Remember, in the scope of this entire week, we already have a pre-trained model, we are not training the model in any way, we are only working with the train model in different ways. So, you provide the input image that you want to your network, and you extract the activations at different layers for the given input image.

Then you also extract what are known as Style Targets. So you give the style image to the same neural network. So, a style image could be a painting by Picasso or Van Gogh, or something like that, you give that as input to the same neural network. But this time, you do not get the activations of this input at different layers. Instead, you get what are known as Gram matrices of ConvNet, activations of all layers. What does that mean?

So, if you take a particular layer here, remember, this is composed of a set of feature maps. So, you have a volume of different set of feature maps. Now, each feature map could be say 20x20 dimensions as an example. Now, what we do here is to linearize or rasterize, this entire feature map into a 400-dimensional vector. And now, you take the covariance of this 400-dimensional vector with the another channel or another feature map in that volume.

And you would now get one such value of the covariance of one channel with the next channel. Similarly, you compute the covariance of every channel with every other channel, and you would get a Gram matrix of ConvNet activations of all layers. Why is this a style target? There is no clear answer. But one could surmise that this is because the relationships between the filters give us an idea of what style is being understood by the neural network. That is the broad idea here.

(Refer Slide Time: 12:33)

Neural Style⁴

1) Extract **input targets** : ConvNet activations of all layers for the given input image.

2) Extract **style targets** : Gram matrix of ConvNet activations of all layers for the given style image.

3) Initialize a new network.

4) Optimize over image to match:
- Activations of **input**.

$\text{argmax}_I S_c(I)$

⁴Gatys et al, A Neural Algorithm of Artistic Style, 2015

Vineeth N.B. (IIT-H) 86.5 Beyond Explaining CNNs 9 / 11

So, what do we do with the style target, once you obtain these Gram matrix of ConvNet activations from the style image, now, you take a new you again, take a network, a train model, and now you give a blank input as an image. And to this image, you backprop the image again, but you try to ensure that the activations that you get at any layer of this model is close to the activations that you get for this image on the same model in a different context.

And the Gram matrix of activations that you get here resemble the gram matrix of activations that you get for the style model. So, remember, this is the content part of it. And this is the style part of it. So, you now backprop to image for a gray image, and another of these Alex Net models or any CNN model, you do backprop. The image, remember that we said was argmax of I . We tried to maximize the score of I with respect to a particular neuron.

But this time, we are going to try to minimize the distance between the activations that we get for this input here, and the activations that we get for this dog image on this network. Similarly,

minimize the distance between the Gram matrix of the covariances between activations here, and the Gram matrix of the covariance between the activations in the style. When you forward propagate the style image.

(Refer Slide Time: 14:12)

Neural Style⁴

NPTEL

Input

Style

- 1) Extract **input targets** : ConvNet activations of all layers for the given input image.
- 2) Extract **style targets** : Gram matrix of ConvNet activations of all layers for the given style image.
- 3) Initialize a new network.
- 4) Optimize over image to match:
 - Activations of **input**.
 - Gram matrix of activations of **style**.

⁴Gatys et al, A Neural Algorithm of Artistic Style, 2015

Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 9 / 11

when you combine these two, you end up getting an image, which takes the style from the style image and the content from the input image. And that is the output that you get while you started with a simple gray image.

(Refer Slide Time: 14:29)

Neural Style: Examples

NPTEL

Credit: Thushan Ganegedara, Intuitive Guide to Neural Style Transfer, TowardsDataScience

Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 10 / 11

Here are some examples of various Neural Styles. So you can see here is the input image, and here is the style and you get a pretty interesting output that resembles the style. So here is Steve

Jobs picture and you have a style and you see the picture changes. Similarly, you see several other examples, which are fairly good in terms of what they achieve.

(Refer Slide Time: 14:56)

Neural Style: Examples

Credit: Artistic Style Transfer with TensorFlow Lite

Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 10 / 11

More examples here, if you would like to take a look at where the content and style is varied, and you have an entire grid of several examples of taking a content and trying different styles on them.

(Refer Slide Time: 15:10)

Homework

Readings

- Sarthak Gupta, DeepDream with Code, HackerNoon
- Thushan Ganegedara, Intuitive Guide to Neural Style Transfer, Towards Data Science
- (Optional) Another good tutorial on Neural Style Transfer on Towards Data Science

Exercises

- Watch this fun video on YouTube of using DeepDream on videos, try to figure out how this was done!

Vineeth N B (IIT-H) §6.5 Beyond Explaining CNNs 11 / 11

So, if you would like to understand this more, there are a couple of interesting links here, which gives you an intuitive understanding of neural style transfer. Both are on the towards data science links. And there is also a nice understanding of Deep Dream on Hacker Noon with the code provided. And an interesting exercise for you is going to be to watch this fun video of YouTube on doing Deep Dream on videos.

The example that we talked about in this lecture was about doing Deep Dream on a given input image. What if you do Deep Dream on videos? That is what this fun video on YouTube does. Please take a look at it. More importantly, try to figure out how it was done. That is the exercise for you this week.