

Deep Learning for Computer Vision
Professor Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture 38
Explaining CNNs - Visualization Methods

Continuing from where we left the last time. This week, we will focus on understanding, visualizing, and explaining the model predictions of a CNN, a convolution neural network, or for that matter any other neural network also. We will start with the first lecture on visualization methods that have different kernels of filters in a CNN or perhaps even activations in a particular layer of a CNN or even other methods that we will see later in this lecture.

(Refer Slide Time: 01:08)



Acknowledgements

- Most of this lecture's slides are based on **Lecture 13 of CS231n: Convolutional Neural Networks for Visual Recognition** course taught by Fei-Fei Li and others at Stanford University
- Some content (AlexNet CNN figures, etc) is also adapted from **Lecture 12 of CS7015: Deep Learning** course taught by Mitesh Khapra at IIT Madras



Vineeth N B (IIT-H)

6.1 Explaining CNNs: Visualization Methods

2 / 15

Most of these lecture slides are based on lecture 13 of CS 231n at Stanford, and some of the content is borrowed from the excellent lectures of Mitesh Khapra at IIT, Madras.

(Refer Slide Time: 01:16)

The slide is titled "Visualize Filters/Kernels (First Layer)¹". It features the NPTEL logo on the left. The main content is divided into several sections: a large grid of 64 filters for AlexNet (64 x 3 x 11 x 11), a 3D visualization of a filter, and smaller grids for ResNet-18 (64 x 3 x 7 x 7), ResNet-101 (64 x 3 x 7 x 7), and DenseNet-121 (64 x 3 x 7 x 7). A section titled "A Closer Look (AlexNet)" shows a detailed view of filters, including one that looks like a Gaussian edge detector. A small video inset shows a man speaking. At the bottom, there is a citation: ¹Krizhevsky, One weird trick for parallelizing convolutional neural networks, 2014. The slide footer includes "Vineeth N.B. (IIT-H)", "6.1 Explaining CNNs: Visualization Methods", and "3 / 15".

Let us start with the simplest form of visualization, which is visualizing the filters or kernels themselves. Remember, that when you have a CNN, in every convolution layer, you have a certain number of filters. For example, if you recall, in the AlexNet, the first convolutional layer had 11x11 filters, it had 96 of them, 48 going to one GPU, and 48 going through the other GPU if you recall AlexNet architecture.

In this particular slide that you see here, we are looking at a variant of AlexNet, which was developed by Alex Krizhevsky a little later in 2014 when he came up with a method to parallelize CNNs, this is just an example to be able to visualize this more easily. So in this variant of AlexNet, the architecture had 64 filters in the first convolutional layer.

So what you see on the left top here is 64 filters, each level across 11x11 in size. And each of them has three channels, the r channel, g channel, and b channel the three colors. So that is what we have is the total number of filters. So, if we visualized each of them on a grid such as this, remember that a filter is an image on its own, just like how convolution is commutative, you can always choose to look at an image as a filter or a filter as an image, it does not matter.

Any matrix of the size of the filter can also be plotted as an image. So, when you do it that way, you get something like what you see on the top right here. So, let us try to look at some of them more carefully. So, if you visualized some of these filters more carefully, you see that there are filters that try to capture oriented edges. So, you can see this one here on the bottom row, the fourth from the left which captures it looks like a Gaussian edge detector, which smoothens out along a certain orientation. Similarly, you have another edge detector here, another edge

detector on top, you also have some which capture slightly higher order variations such as a checkerboard kind of a pattern or a series of striations so on and so forth. You also have color-based edge detectors. So, in the last filter here on the bottom right, you see an edge detector that goes from green to pinkish or red color. So, you see similar color-based filters even on the top left here.

So does is this a characteristic of AlexNet alone? Not, if you took even the filters of resnet 18, resnet 101, or densenet, 121, In each of these, the filters in the first convolutional layer have a very similar structure. All of them detect edges of different orientations. Certain higher-order interactions, such as checkerboard patterns, striations in different orientations, color blobs, certain color gradations as in edges in different colors, so on and so forth.

You will see this as part of the assignment this week where you try out some of these experiments. So this tells us that the first layer seems to be acting like low-level image processing, edge detection, blob detection, maybe a checkerboard detection, so on and so forth. Remember here, that these are filters that are completely learned by a neural network, which we did not prime in any way.

(Refer Slide Time: 05:32)

Visualize Filters/Kernels (First Layer)

- You can visualize the kernels of higher layers but it is just not interesting
- Input to higher layers is no more the images we know or understand, so becomes difficult to interpret the filters beyond the first layer

Weights: $16 \times 3 \times 7 \times 7$ (layer 1 weights)

Weights: $20 \times 16 \times 7 \times 7$ (layer 2 weights)

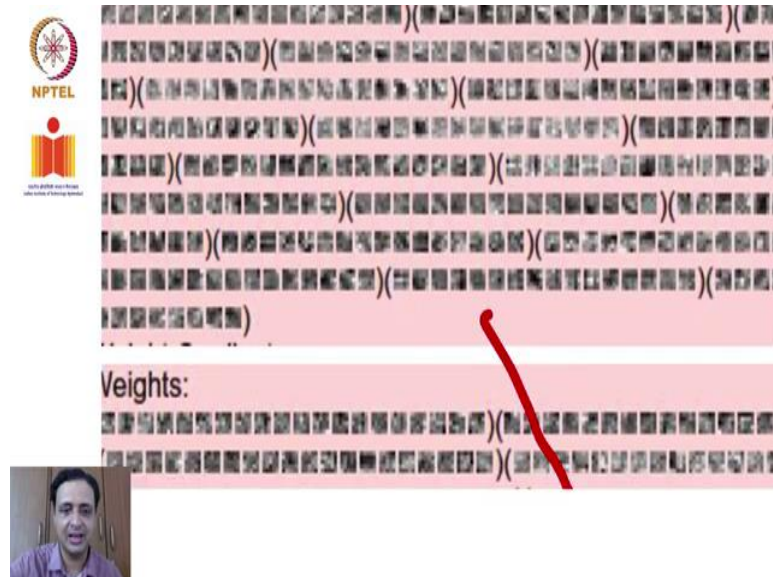
Weights: $20 \times 20 \times 7 \times 7$ (layer 3 weights)

Vineeth N B (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 4 / 15

You can also visualize the kernels of higher layers, just like how we did it for the first convolutional layer. You could also take all the filters of the second convolutional layer, the third convolutional layer, so on and so forth. But it happens that if you had to generalize them across applications, they are not that interesting. We did see an earlier example last week, where we took face images and showed that filters in the first layer correspond to low level image

features. Then we talked about middle layers, extracting noses, and eyes, and so on. And then we talked about the later layers extracting face-level information. It does happen in certain applications but in general, if you had a wider range of objects, if you only focused on faces or a smaller group of objects, maybe you could make sense of the higher layers, filters.

(Refer Slide Time: 6:52)



But in a more general context such as image net, which has a thousand classes in the data set. These kinds of visualizations of filters of higher layers are not that interesting. So here are some examples, so you can see that the weights, remember in a CNN, the weights are the filters themselves.

So if you look at weights in a later layer, you see that it may not be that interesting for understanding what a CNN is learning. That is because of the variety of classes that may result in various obstructions across the data set. So the input to the higher layers is no more the images that we understand. At the input layer, we know what we are providing as input but when you go to higher layers, you do not know what you are providing as input, so it becomes a little bit more difficult to understand what is happening.


(Refer Slide Time: 07:21)

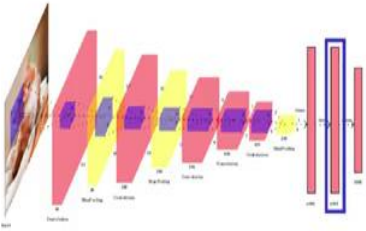
The slide is titled "Visualize Filters/Kernels (First Layer)" and contains the subtitle "The Gabor-like filters fatigue". It features several grids of grayscale images representing learned filters. On the left, there are logos for NPTEL and an open book icon. At the bottom left, there is a small video inset of a man speaking. At the bottom, a footer contains the text "Vineeth N B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 5 / 15".

However, if you take the filters of the first layer alone, across various models and datasets, Hope by now you are familiar with the various CNN models such as AlexNet, ResNet, dense net, VGG, so on and so forth. So if you have to look at the filters of the first layer, first convolutional layer, across all of these models you get very similar kinds of filters and it is generally called the Gabor-like filters fitting. Why is that so? Recall the Gabor filters discussion that we had earlier in the course, where we said, a Gabor filter is like a combination of a Gaussian and a sinusoid.

So, and you can change the scale and you can change the orientation of the Gabor filter. And you end up detecting edges in different orientations, you perhaps end up detecting different striations, checkerboard patterns, so on and so forth, which is exactly what we see as the CNN learning on its own too. So that is the reason why we call this entire visualization of the filters of the first convolutional layer a Gabor, like filter fatigue, by fatigue here we just mean, it is exactly the same across all of these models and data sets.


(Refer Slide Time: 08:45)

 **Visualize the Representation Space (Last Layer)**



- 4096-dimensional feature vector for an image (layer immediately before the classifier)
- Run the network on many images, collect the feature vectors


Vineeth N B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 6 / 15



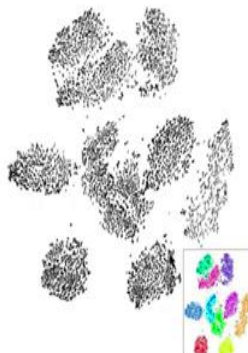
Another option, other than visualizing the filters in different layers, when we talk about visualizing the filters, remember it is 11×11 or a 7×7 or whatever be the size of the filter, you simply have to plot it as an image. Another thing that you can do is to visualize the representation space learned by CNN. What do we mean, if you took the AlexNet, remember that the output of FC 7 or the fully connected layer in the 7th position of the depth of the network, which we denote as FC 7 is a 4096-dimensional vector?

That is the layer immediately before the classification. So what we can do is take all the images in your test set or validation set for that matter and you forward propagate those images until this particular layer and collect all these 4096-dimensional vectors.

(Refer Slide Time: 9:48)


 **Visualize the Representation Space (Last Layer)**

- Visualize the "space" of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions
- Use any dimensionality reduction algorithm
- Simple algorithm: Principal Component Analysis (PCA)
- More complex: **t-SNE** (right)



van der Maaten and Hinton, Visualizing High-Dimensional Data Using t-SNE, Journal of Machine Learning Research, 2008

Vineeth N B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 7 / 15



What do we do with them? You can now visualize the space of these FC 7 feature vectors by reducing the dimensionality from 4096 to any dimension of your choice. But for simplicity, let us say two dimensions. How do we do this? Once again, hopefully, you have done a basic machine learning course.


And you know that you can use any dimensionality reduction method to be able to do this. A simple example could be principal component analysis. So you take all of those 4096-dimensional vectors of several input images, and you do a PCA on top of them to bring all of them into two-dimensional space.

A more popular approach, which is considered to be a very strong dimensionality reduction method is also t-SNE, which stands for t Stochastic Neighborhood Embedding, this was a method once again developed by Hinton along with Vonda Martin in 2008. We also have a link for this towards the end of this lecture.


So you can play around with the t-SNE if you would like to understand it more. So when you apply t-SNE, on the representations that you get as the output of the CNN in the penultimate layer, you end up getting results such as this, this is specific for the enlist data set, where you have 10 classes, this is the handwritten digit data set. So you see here that each class invariably goes to its cluster.

This seems to tell us while we cannot, in reality, visualize a 4096-dimensional space. By bringing it down to two dimensions, we understand that the classes, the representations belonging to different classes are fairly well separated into different clusters. And why is that important? Now, developing a classification algorithm on these representations becomes an easy task. And that is why having a classification layer right after that penultimate layer of representations makes the entire CNN work well to classify.


(Refer Slide Time: 11:58)

**t-SNE Visualization**

- Images that are nearby each other are also close in the CNN representation space, which implies that the CNN "sees" them as being very similar



- Notice that the similarities are more often class-based and semantic, rather than pixel and color-based.

 Vineth N B. (IIT-H) 6.1 Explaining CNNs: Visualization Methods 8 / 15

So here is an example of the same for image net. So where this is a plot, a two-dimensional plot of various images in the image net data set, taken to 4096-dimensional space by AlexNet, and then brought down to two-dimensional space and plotted on a two-dimensional map, the only thing we are doing here is, we are putting the respective image on each location, just for understanding what is happening. So this is just a huge map, and if you had to look at one particular part of it, let us try to zoom in into one particular part of it.

(Refer Slide Time: 12:42)

 es that are nearby each other are also close as that the CNN "sees" them as being very similar





You see, that all the images corresponding to say a field seem to be coming together in this space of representations. For this matter, if you scroll around and see other parts of it, you will see that similar objects you can see at many points of these embeddings here that at many

points, you see very similar objects being grouped, you see all cars somewhere here, so on and so forth. This tells us that these embeddings or representations that you get out of the penultimate layer, actually capture the semantic nature of these images. And objects that have similar semantics are grouped, while objects of different semantics are far apart from each other. So this gives us an understanding the CNN representations seem to be capturing the semantics.

Keep this in mind, when we talked about handcrafted features and learn representations, this is what we were talking about. In handcrafted features such as Sift or Hog or LBP, you have to decide what may be useful for a given application and then hand design that filter that you want to use as a representation of the image after which you may apply a machine learning algorithm. But now, we are letting the neural network the CNN in particular, automatically learn these representations that it needs to solve a particular task.

(Refer Slide Time: 14:16)

The slide, titled "Visualize Activations", features the NPTEL logo on the left. The main content area shows a grayscale input image of two people's faces on the left and a 128x13x13 grid of grayscale feature maps on the right. A text box above the grid states: "Conv5 feature map is 128 x 13 x 13; Visualize as 128 13 x 13 grayscale images". Below the grid, a small caption reads: "Figure copyright: Jason Yosinski, 2014." The footer contains a video thumbnail of a speaker, the text "Vineeth N B (IIT-H)", "6.1 Explaining CNN: Visualization Methods", and the slide number "9 / 15".

Here is a visualization of the Conv5 feature maps, in AlexNet the Conv5 feature map is 128x13x13. So there are 128 feature maps each 13 by 13. If you now visualize them as grayscale images, you can see something interesting here. So when this specific image with two people is given as input, you see that one of the filters, or actually, there are quite a few of them in fact, seem to be capturing the fact that there are two entities in the image. So this could give you a hint that the later layers in the CNN can capture these higher-level semantics of the objects in the images.

(Refer Slide Time: 15:00)

The slide features the NPTEL logo on the left. The main title is "Visualize Maximally Activating Image Patches". Below the title is a bullet point: "Consider a CNN, and a single neuron in any of its intermediate layers". The central diagram illustrates a CNN architecture with an input image, followed by several convolutional layers (labeled Conv1, Conv2, Conv3, Conv4, Conv5) and fully connected layers (FC1, FC2, FC3). A specific neuron in the Conv3 layer is highlighted in green. To the right of the diagram are three vertical bars representing the output of the fully connected layers. At the bottom left is a video feed of the speaker, and at the bottom right is a footer with the text "Vineeth N B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 10 / 15".

Another way of visualizing and understanding CNNs is, you could extend the same thought and consider a trained CNN. Remember that all of this analysis is for a trained CNN, after training the CNN, you want to understand what it has learned. Remember, that is the context in which we are talking about this.

So you can consider a CNN, and consider any single neuron in its intermediate layers, so let us consider that particular one in green. Now, you can try to visualize which images cause that particular neuron to fire the most. So you can give different images as input to the CNN, and keep monitoring that particular neuron and see which image is making it fire the most. What can we do with that?

Now we can work backward and understand that this particular pixel here will have a certain receptive field in the previous convolutional layer, which means that is the region that led to this pixel being formed in this particular convolutional layer. Similarly, you can take the pixels in the previous layer, and look at the receptive field of each of them and find the receptive field in the earlier layer, in this case, the first convolutional layer, you can go further again and find out the receptive field in the original image, which was responsible for this pixel in the third convolutional layer.

Remember, we were also discussing this when we talked about backpropagation through CNNs, where we try to understand the receptive field that leads or to a particular pixel getting affected in a particular layer. It is the same principle here.

(Refer Slide Time: 16:56)

The slide features a title bar with the text "Visualize Maximally Activating Image Patches" in red. Below the title, there is a bullet point: "Repeating this for others neurons in the CNN shows us a pattern". The main content is a large grid of approximately 60 small images, arranged in 6 rows and 10 columns. The first row shows various busts of people. The second row shows different breeds of dogs. The third row shows various honeycomb patterns. The fourth row shows a US flag. The fifth row shows a red blob. The sixth row shows specular reflections. At the bottom of the slide, there is a small video feed of a man speaking, and a footer with the text "Vineeth N B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 11 / 15".

Now, if we took images, and try to understand which of them caused a particular neuron to fire, we end up seeing several patterns. So in this set of images, each row corresponds to one particular neuron that was fired, and the set of images and the region inside the set of images, which is shown as a white bounding box, which caused that neuron to fire.

So here is the first row, interestingly all of those images correspond to people, especially busts of people, it looks like that particular neuron was capturing people until they are a bust until their chest. The second neuron here seems to be capturing different dogs, maybe some honeycomb kind of a pattern, even a US flag was taught that honeycomb kind of a pattern is present.

A third neuron captures a certain red blob across all of the images. The fourth neuron captures digits in images, and if you look at the last neuron here, the sixth row, you see that it seems to be capturing specular reflection in all of the images. So over time, as you train the CNNs, each neuron is getting fired for certain artifacts in the images.

And this should probably go back and help you connect to drop out, where we try to ensure that no particular neuron or weight overfits to a training data. And we allow all neurons to learn a diverse set of artifacts in images. So this should help you connect to drop out in that sense.

(Refer Slide Time: 18:40)

 **Visualize Maximally Activating Image Patches**

- Repeating this for others neurons in the CNN shows us a pattern




Springenberg et al. "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015.
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller

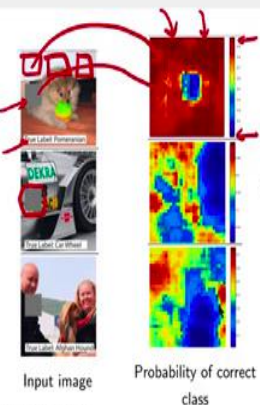
Vineeth N.B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 11 / 15

Here are further examples of the same idea where you take different neurons in CNN and try to see which images or patches of images fired that neuron the most. Once again, you see a fairly consistent trend here, that some of them seemed to fire for I think, this is an eye of an animal, there is again some text in images, there is vertical text and images, there are faces, there are dog faces, again, and so on.

(Refer Slide Time: 19:12)

 **Occlusion Experiments²**

- Typically, we are interested in understanding which portions of an image are responsible for maximizing probability of a certain class
- Occlude (gray out) different patches in the image (centered on each pixel), and see effect on predicted probability of the correct class \Rightarrow gives you a probability for each pixel



Input image | Probability of correct class

²Zeiler and Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
Vineeth N.B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 12 / 15

And the last method that we will talk about in this lecture is what is known as occlusion experiments, which attempt to leverage our objective that we finally want to understand which pixels in an image corresponding to the object recognized by the CNN. Why does this matter? We would like to know if the CNN looked at the cat in the image, while giving the label as a

cat for the image, or did it look at a building in the background or grass on the ground? Remember, a neural network learns correlations between various pixels that are present in your data set to be able to give good classification performance. So if all of your cats in your data set were found only on grass, a neural network could assume that the presence of grass means a presence of a cat.

If you have a test set, where a cat is found in a different background, the neural network may now not be able to predict that as a cat. So to be able to get that kind of trust in the model, the model was indeed looking at the cat while calling it a cat. The occlusion experiments, do this using a specific methodology.

So given these images that you see here, we occlude out different patches in the image centered on each pixel. And you see the effect on the predicted probability of the correct class. Let us take an example, so you can see a gray patch here on the image so you occlude that part of the image, fill it with gray, and send the whole image as input to the CNN, and you would get a particular probability for the correct label in this case, which is Pomeranian. So that is plotted as a probability in that particular location.

Similarly, you gray out a patch here, send the full image as input to a CNN, get a probability for a Pomeranian. How do you get the probability as the output of the softmax activation function, and that probability value is plotted here in this image. So by doing this, by moving your gray patch across the image, you will have an entire heat map of probabilities of whether a pixel or a patch around a pixel reduces the probability of a Pomeranian or does it keep it the same way.


So in this particular heat map, red is the highest value, and blue is the lower value. So you notice here, that when the patch is placed on the dog's face, the probability of the image being a Pomeranian drops to a low value. So this tells us that the image for the CNN model was in fact, looking at the dog's face to call this a Pomeranian.

So in fact, this entire discussion came out of Zeiler's and Fergus' work on visualizing and understanding convolutional neural networks. It is a good read for you to look at the end of this lecture. They observe that when you place a gray patch on the dog's face, the class label predicted by the CNN is a tennis ball, which is perhaps the object that takes precedence when you cover the dog's face. Similarly, in the second image, you see that the true label is the car

wheel, and as you keep using this gray patch all over the image, you see that the probability drops to the lowest when the wheel is covered.

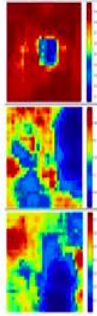

And the third image more challenging is where you have two humans and a dog in between them, which is an Afghan hound, which is the true label, you once again see that when the pixels corresponding to the dog are occluded by the gray patch, the probability drops for the Afghan hound drops low. This is even trickier because there are humans and the model could have been biased or affected by the presence of those humans, but the model does well in this particular case.

(Refer Slide Time: 23:47)




Occlusion Experiments²

- Typically, we are interested in understanding which portions of an image are responsible for maximizing probability of a certain class
- Occlude (gray out) different patches in the image (centered on each pixel), and see effect on predicted probability of the correct class \implies gives you a probability for each pixel
- For example, the first heat map (top) shows that occluding the face of the dog causes a maximum drop in the prediction probability



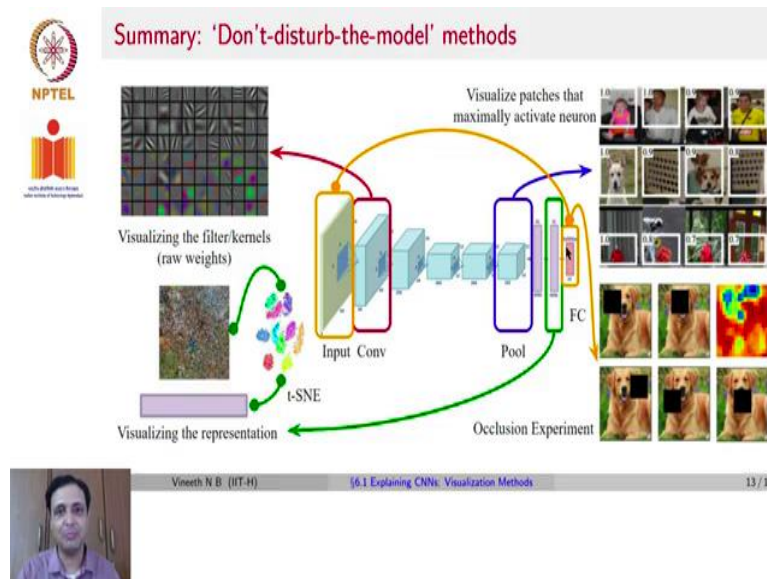
Input image Probability of correct class

²Zeiler and Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
Vineeth N B (IIT-H) 6.1 Explaining CNN: Visualization Methods 12 / 15



So occluding, the face of the dog causes a maximum drop in the prediction probability.

(Refer Slide Time: 23:51)



To summarize the methods that we covered in this lecture, given a CNN, we are going to call all of these methods as do not disturb the model methods, which means we are not going to touch anything in the model, we are only going to use the model as it is and be able to leverage various kinds of understanding from the model.

So you can take the convolutional layer, and then visualize your filters and kernels that is one of the first methods that we spoke about. Unfortunately, this is only interpretable at the first layer, and may not be interesting enough at higher layers. And we also talked about the Gabor-like filter fatigue here. You could also take any other neuron, for example, in a pooling layer, and visualize the patches that maximally activate that neuron, you can get some understanding of what the CNN is learning using this kind of approach.

The third thing that we talked about is you can take the fully connected layer, the representations that you get at the fully connected layer, and visualize the representation and then do dimensionality reduction methods such as t-SNE on these representations, and you get an entire set of embeddings for image net.

And lastly, we spoke about the occlusion experiment, where you take input and perturb the input, and then see what happens at the final classification layer and that gives you a set of a heat map that tells you which part of the image the model was looking at while making a prediction.

(Refer Slide Time: 25:37)



Readings

- Summary of Visualizing CNNs
 - [Lecture Notes of CS231n, Stanford](#)

Miscellaneous



- Deep Visualization Toolkit [demo video](#) and [webpage](#) by J. Yosinski et al.
- To see high-resolution t-SNE visualizations, visit [here](#)
- To know more about t-SNE, visit [here](#)



Vineeth N.B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 14 / 15


Recommended readings, lecture notes of CS 231n, as well as a nice deep visualization toolkit demo video on the web page by Jason Jasinski, I would advise you to look at that. You can also get to know more about t-SNE and t-SNE visualizations as a dimensionality reduction technique on these links provided here.

(Refer Slide Time: 25:53)



References

- Laurens van der Maaten and Geoffrey E. Hinton. "Visualizing Data using t-SNE". In: 2008.
- Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Nov. 2013).
- Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *ECCV*. 2014.
- Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *CoRR* abs/1412.6806 (2015).



Vineeth N.B. (IIT-H) | 6.1 Explaining CNNs: Visualization Methods | 15 / 15

Here are some references.