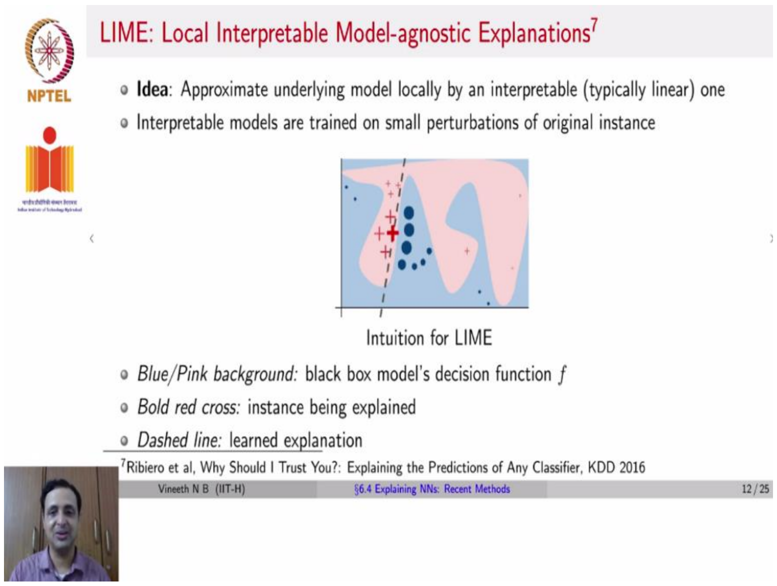
Deep Learning for Computer Vision
Professor Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Explaining CNNs: Recent Methods
Part 02

(Refer Slide Time: 00:13)



LIME: Local Interpretable Model-agnostic Explanations[7]

- **Idea**: Approximate underlying model locally by an interpretable (typically linear) one
- Interpretable models are trained on small perturbations of original instance

Intuition for LIME

- *Blue/Pink background:* black box model's decision function f
- *Bold red cross:* instance being explained
- *Dashed line:* learned explanation

[7]Ribiero et al, Why Should I Trust You?: Explaining the Predictions of Any Classifier, KDD 2016

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 12 / 25

A popular approach for explainable machine learning in general, not just Neural networks, not just CNNs, is known as Lime, Lime stands for Local Interpretable Model Agnostic Explanations. So, each of those words means something which should become clearer when we explain the method. The way the method works is to approximate any underlying model by an interpretable linear model.

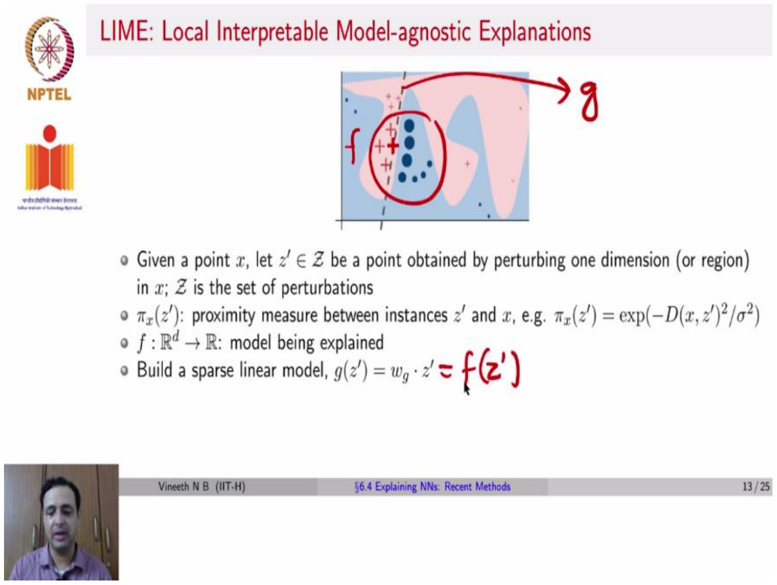
Let us see how it does that. So, you first have a trained model. And now you are going to train an interpretable model on top of some small perturbations of a given instance. So here is an intuitive visualization of the line approach. So the blue and pink background here gives the black box models decision function f. So that is the classification boundary that you have between two classes, let us assume it is a binary classification problem.

And all of these pink regions correspond to one class, all of these blue regions correspond to the other class. The Bold Red Cross is one specific test instance that you are given, which you have to explain. So, I have been given an image for instance of a cat. And I know how to explain. And let us assume that this is the decision boundary of a CNN, I now have to explain why this Red Cross is called a cat.

So that is what, that is what we have been talking about. So far, this is just a different way of viewing the same problem. And finally, you also have a dashed line here, which is a learned explanation. Now, let us see how this explanation is learned. The way it is learned is to take this bold red cross, which is a given image, construct a few perturbations of the image. So, you could not take an image, occlude a part of it, occlude a segment of it, occlude an object in it, whatever it be, these are different kinds of perturbations that you construct around the image.

And you notice that when you remove a certain portion, our model classifies it as positive. And when you remove another portion, our model classified it as, classifies it as negative. And that is why you see these plus reds and these blue dots, which means those respective perturbations were classified by our model as belonging to class one or class two. So, you should be able to intuitively understand now how this can be used. If I know which perturbations are making me lose the class label and make it go to another class, you know that they are my most important regions in the image, too, for a particular class label to be assigned to that image.

(Refer Slide Time: 03:10)



LIME: Local Interpretable Model-agnostic Explanations

- Given a point x, let $z' \in \mathcal{Z}$ be a point obtained by perturbing one dimension (or region) in x; $\mathcal{Z}$ is the set of perturbations
- $\pi_x(z')$: proximity measure between instances z' and x, e.g. $\pi_x(z') = \exp(-D(x, z')^2/\sigma^2)$
- $f : \mathbb{R}^d \rightarrow \mathbb{R}$: model being explained
- Build a sparse linear model, $g(z') = w_g \cdot z' = f(z')$

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 13 / 25

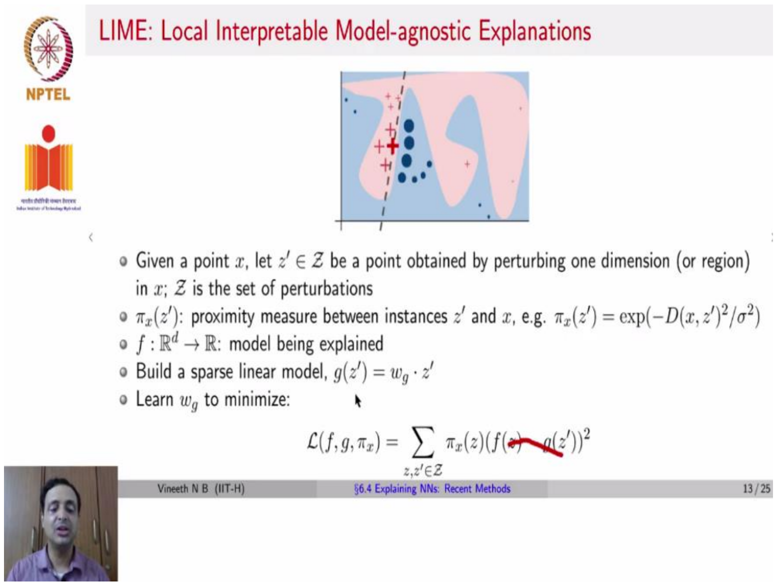
Let us see this in more detail. So given a point x, that is your bold red cross let z' be a point obtained by perturbing one dimension or region in x. So, you could perturb up a pixel, or you could perturb up a particular region, you could do occlusion kind of experiments, or you could just perturb up an entire region of an image. And Z is the set of all perturbations. Let us assume that we have some metric of $\pi_x(z')$, which is a proximity measure between instances z' and x.

So, for example, if you take this the not bold plus, and the bold plus the distance between them, is what we would refer to as $\pi_x(z')$, this could be if a distance metric, such as an inverse exponential distance metric. So you could have an exponent of minus d exact prime whole square, where d is a squared distance divided by some sigma square, which can be predefined by a given user. f is again, the model is explained. Our goal now is given these perturbations $z'$and your original image x, we want to build a model $g(z')$'s such that $g(z') = w_g z'$, which is the linear model, such that the output of $g(z')$ is the same as the output of $f(z')$.

So This means we want this to be also equal to $f(z')$. What do we mean? Remember that f is this function that is already given to us. G is this linear model that we are trying to develop. using only those perturb examples, and we are saying now, that g should be learned in the neighborhood of x in such a way that it makes the same decisions on all of these perturbations that the original model would, on those in if you gave those as inputs.

So, in a sense, g is like a local approximation of f, f is what occurs across the domain of the inputs. If you take a particular example, and perturbations of that example around that around its neighborhood, then g would be a linear model that approximates the original decision surface of f in that local neighborhood.

(Refer Slide Time: 05:42)



LIME: Local Interpretable Model-agnostic Explanations

- Given a point x, let $z' \in \mathcal{Z}$ be a point obtained by perturbing one dimension (or region) in x; $\mathcal{Z}$ is the set of perturbations
- $\pi_x(z')$: proximity measure between instances z' and x, e.g. $\pi_x(z') = \exp(-D(x,z')^2/\sigma^2)$
- $f : \mathbb{R}^d \rightarrow \mathbb{R}$: model being explained
- Build a sparse linear model, $g(z') = w_g \cdot z'$
- Learn w_g to minimize:

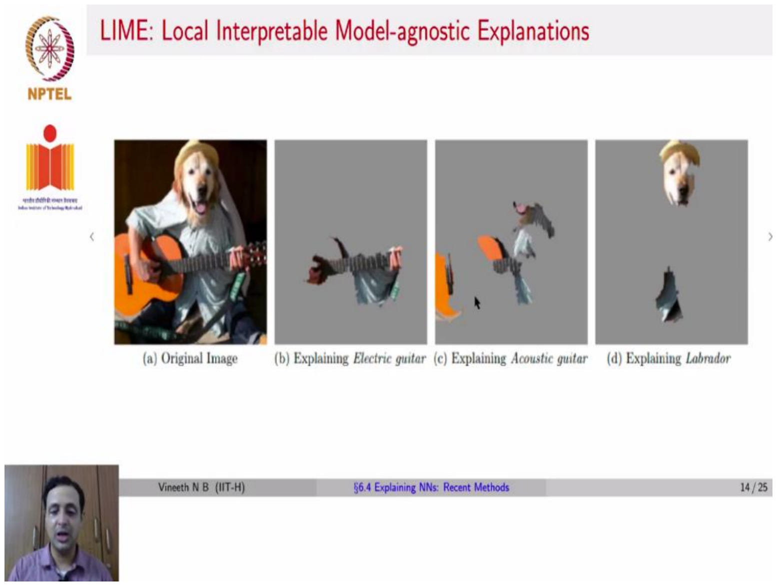
$$\mathcal{L}(f,g,\pi_x) = \sum_{z,z' \in \mathcal{Z}} \pi_x(z)(f(z) - g(z'))^2$$

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 13 / 25

Now, we want to learn w_g, that would be our goal, because we want to learn that linear model, w_g, is learned by minimizing, you take two z and z's belonging to z. $\pi_x(z)$, which is the distance of x from z, $f(z) - g(z')$. So, you want to ensure that the decisions of f, as well as the decision of g, are together, they make the same prediction. And you also want to keep in mind how far z is from x.

So, if it is very far away, you can perhaps allow it to make a little bit of a mistake. But if it is close by, you do not want this, you do not want to lose anything in the decision at all. That is the last function that we use to learn g of z prime.

(Refer Slide Time: 06:38)



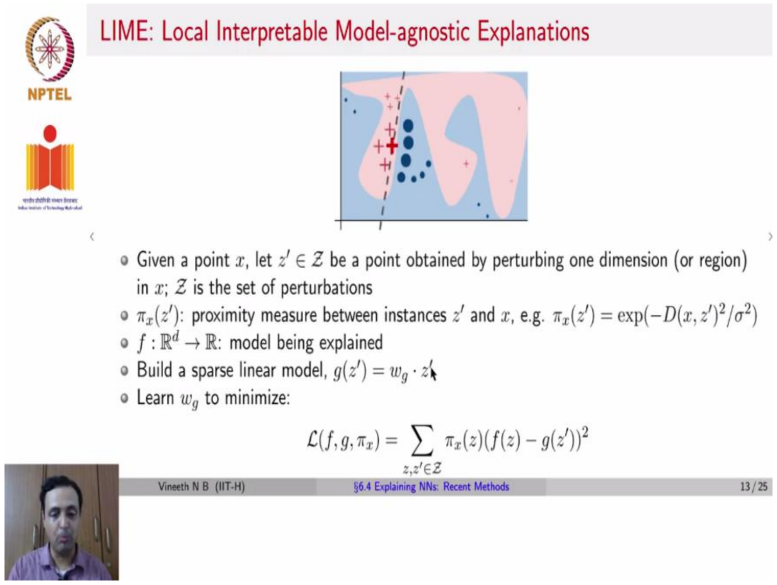
LIME: Local Interpretable Model-agnostic Explanations

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 14 / 25

Let us see how this is used. So given an image such as this, your perturbations could be where you take to let us see the face of the dog and remove it and give the rest of the image and see what the network predicts as similarly, you may receive, you may remove the strings in the guitar and see what the model predicts as similarly then you remove your, the board of the guitar, and you would see what the model predicts says. And based on all of these, if you learn a g,

(Refer Slide Time: 07:11)



LIME: Local Interpretable Model-agnostic Explanations

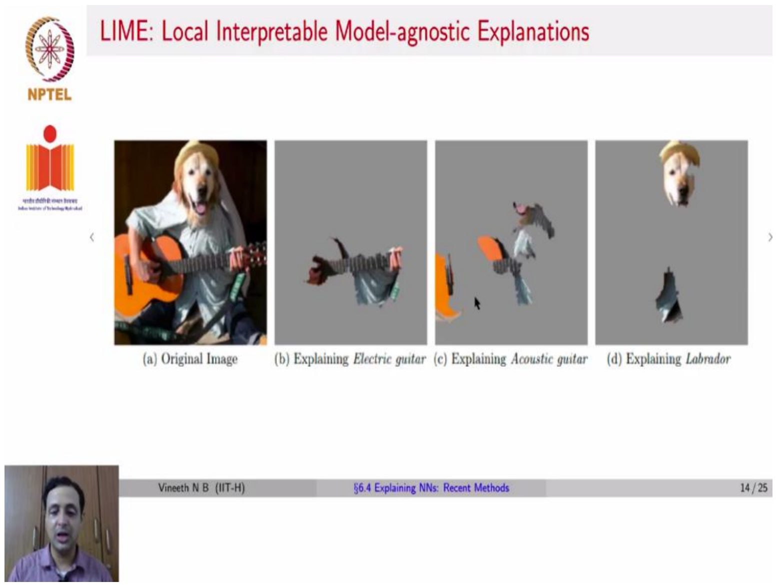
- Given a point x, let $z' \in \mathcal{Z}$ be a point obtained by perturbing one dimension (or region) in x; $\mathcal{Z}$ is the set of perturbations
- $\pi_x(z')$: proximity measure between instances z' and x, e.g. $\pi_x(z') = \exp(-D(x, z')^2/\sigma^2)$
- $f : \mathbb{R}^d \rightarrow \mathbb{R}$: model being explained
- Build a sparse linear model, $g(z') = w_g \cdot z'$
- Learn w_g to minimize:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in \mathcal{Z}} \pi_x(z)(f(z) - g(z'))^2$$

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 13 / 25

The way we talked about it, you would get your w_g-s, which would tell you how much each of these z'-s weights towards these decisions. In a sense, that tells you which of these z'-s is most important for a particular decision.

(Refer Slide Time: 07:27)



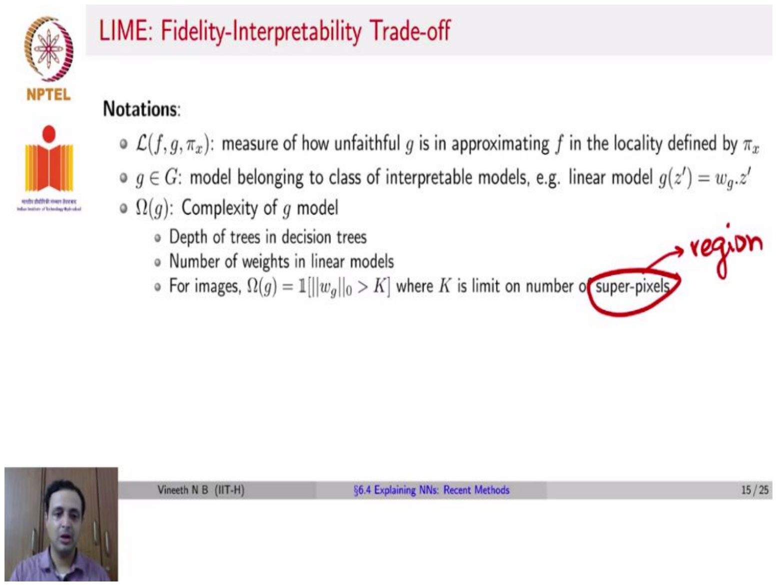
LIME: Local Interpretable Model-agnostic Explanations

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 14 / 25

Rather, you would be able to say that these are the regions in the image that are most important to call this image as corresponding to electric guitar. These are the regions that are most important to call this image as corresponding to acoustic guitar. And these are the regions corresponding to this image, which correspond to the label Labrador.

(Refer Slide Time: 07:53)



LIME: Fidelity-Interpretability Trade-off

Notations:

- $\mathcal{L}(f, g, \pi_x)$: measure of how unfaithful g is in approximating f in the locality defined by π_x
- $g \in G$: model belonging to class of interpretable models, e.g. linear model $g(z') = w_g.z'$
- $\Omega(g)$: Complexity of g model
  - Depth of trees in decision trees
  - Number of weights in linear models
  - For images, $\Omega(g) = \mathbb{1}[||w_g||_0 > K]$ where K is limit on number of super-pixels

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 15 / 25

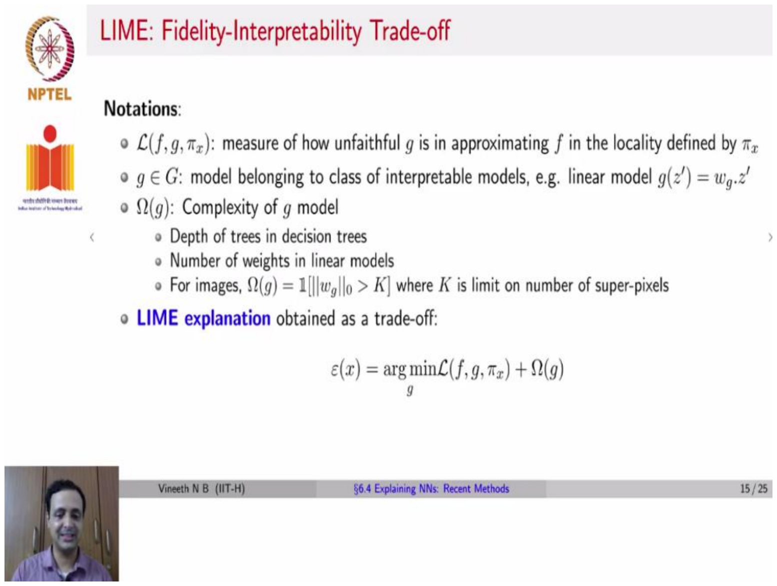
Lime also has a fidelity slash x interpretability trade-off. And the reason for this is the last function that we used, tries to ensure that the function g, the linear model that we learn should be approximating f in the locality defined by $\pi(x)$. So the goodness of function g is very important. At the same time, you also do not want g to be too complex a function. So g ideally, is a model that belongs to the class of all interpretable models.

So a simple example would be a linear model. And we define $\Omega(g)$ to be the complexity of the model. What do we mean by complexity, if we are using g to be a decision tree, it could be a depth of the tree. If we are using a linear model, it would be the number of nonzero weights. If we are using it for image-based analysis, we may simply say that the number of superpixels should not be greater than a certain amount.

What is a superpixel? Region. A superpixel is just a region of an image, we would not want more than three to four regions to be nonzero in that model when you make your decision. Remember, in this case, that g is a specific model learned for only one image, that bold red cross that we saw, if you gave a different image, you will have to learn a different g. So this is the complexity of the g model.

In the earlier example that we talked about a couple of slides back, we learned a Sparse Linear model, you could use, say, L1 regularization to make your model sparse, as you may have learned in machine learning. That is what we use here to be able to get these models to be sparse.

(Refer Slide Time: 09:45)



LIME: Fidelity-Interpretability Trade-off

Notations:

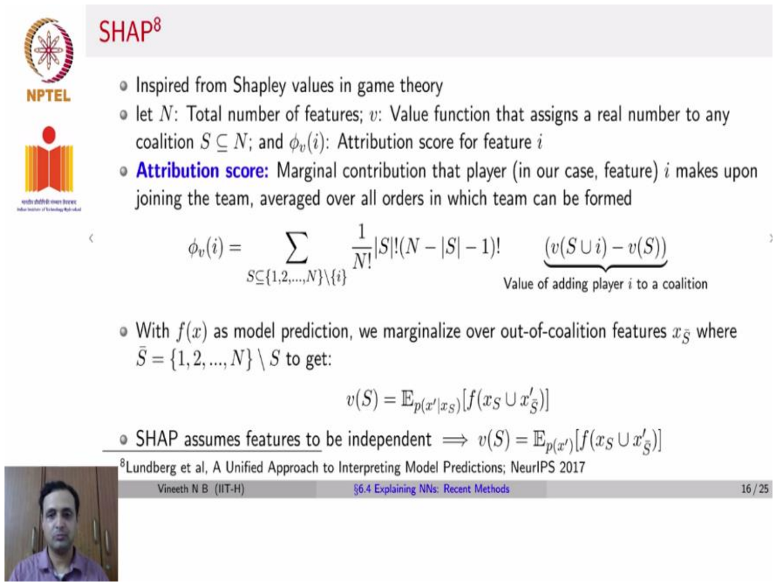
- $\mathcal{L}(f, g, \pi_x)$: measure of how unfaithful g is in approximating f in the locality defined by π_x
- $g \in G$: model belonging to class of interpretable models, e.g. linear model $g(z') = w_g.z'$
- $\Omega(g)$: Complexity of g model
  - Depth of trees in decision trees
  - Number of weights in linear models
  - For images, $\Omega(g) = \mathbb{1}[||w_g||_0 > K]$ where K is limit on number of super-pixels
- **LIME explanation** obtained as a trade-off:

$$\varepsilon(x) = \arg\min_g \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 15 / 25

So now why does that matter? You could now view explanations in line as a tradeoff between the fidelity which is the fidelity of $g(f)$, which is given by the first term, and the complexity or the interpretability of the explanation, which is the second term, it may not be possible for both of these to be minimized. If you want g to be truer to f, you may need a more complex g. But if you make a more complex g, the interpretability could be reduced. So, it could lead to this kind of a trade-off. But irrespective if you chose a linear model for g, it does work fairly well in various settings.

(Refer Slide Time: 10:26)



SHAP[8]

- Inspired from Shapley values in game theory
- let N: Total number of features; v: Value function that assigns a real number to any coalition $S \subseteq N$; and $\phi_v(i)$: Attribution score for feature i
- **Attribution score:** Marginal contribution that player (in our case, feature) i makes upon joining the team, averaged over all orders in which team can be formed

$$\phi_v(i) = \sum_{S \subseteq \{1,2,\ldots,N\} \setminus \{i\}} \frac{1}{N!}|S|!(N-|S|-1)! \underbrace{(v(S \cup i) - v(S))}_{\text{Value of adding player } i \text{ to a coalition}}$$

- With $f(x)$ as model prediction, we marginalize over out-of-coalition features $x_{\bar{S}}$ where $\bar{S} = \{1,2,\ldots,N\} \setminus S$ to get:

$$v(S) = \mathbb{E}_{p(x'|x_S)}[f(x_S \cup x'_{\bar{S}})]$$

- SHAP assumes features to be independent $\implies v(S) = \mathbb{E}_{p(x')}[f(x_S \cup x'_{\bar{S}})]$

[8]Lundberg et al, A Unified Approach to Interpreting Model Predictions; NeurIPS 2017

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 16 / 25

The last model that we will see here is inspired by what is known as Shapley values in Game Theory. This method is called SHAP and was introduced in NeurIPS of 2017. To understand this model, let us define a few notations. Let N be the total number of features in our, in our method in our setting. Let small v be a value function similar to attribution. But we will see there is a slight difference. A value function that assigns a real number to any coalition S subset of N.

By coalition, we mean a subset of features. Why are we calling it a coalition? Because this is inspired by Game Theory. So, an example where Shapley values can be used is imagined a football team. And you want to find out how important Messi is for, say the Barcelona team. So, you want if you want to find that out, you have a team already, let us say five to six players. And you see if you added Messi to that, what value would the team now obtain with Messi and without Messi with the other five players.

So that is so now, the team is what we refer to as coalition. So, in game theory could be players or any other context. But for us, each of those are attributes we are saying that if we already have five attributes if we now add a sixth attribute, how much is our value changing? That is what we want to measure. And phi sub v of i is the attribution score for a feature i.

Given these notations. attribution score in shap in the shap method is given by the marginal contribution, that player in our case, a feature or an attribute or a dimension, whatever you want to call it. A marginal contribution that player i makes upon joining the team averaged overall orders in which the team can be formed. Let us let us again take the example of Messi. So if you wanted to understand what value, Lionel Messi brings to the Barcelona football team, you consider the other players in the team, and you consider them in all possible permutations.

And see if Messi is added how the value changes. So you take the first three players add Messi, see how the value changes, you take seven players add Messi see how the value changes, you take seven different players to add Messi to see how the value changes. Average the value increase that you got across all of these possible subsets. And that is the value that Messi would bring to the Barcelona team.

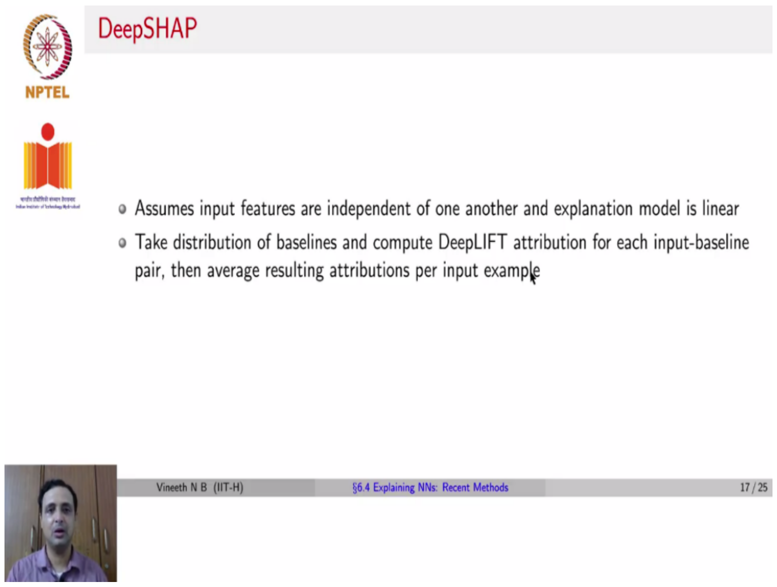
Formally, you would write this as $\phi_v(i)$ is equal to summation over all the subsets without Messi without i, $\frac{1}{N!}|S!|(N-|S|-1)!$ this looks at all possible combinations in the subset without Messi given N to be the total number of players or features for us. And this term here says $v(S \cup i) - v(S)$, what if you add i to the team S minus only the team S without i.

That is the value of adding player i to a coalition. So, I hope you could translate this to understanding the usefulness of a feature in a Neural network. But we have to answer one question, what is v for a neural network? What is that value? How do you measure that? So, if you have a function f, which is a neural network, which is for predicting, then we are going to define v as v of S, which is there S is a subset of features is given by.

The expectation of $p(x')$ given x_S,$f\left(x_S \cup x'_{\bar{S}}\right)$: $\bar{S}$ is the complement of S. Rather we are seeing now that the value of a set S is given by expectation over expectation over all possible probabilities of picking elements x'given subset access already given a team of or a set of features S already $f(x_S \cup x'_{\bar{S}})$. What would be the output if you added an $x'$from $\bar{S}$ to x_S.

So, you take all of those f's outputs to take the expectation, that becomes the value of S here. But in this particular method of SHAP, we assume that the features are independent, which means this conditional probability simply becomes the probability of picking a particular feature from $\bar{S}$, which is given by x'. So, you try to see, given an x_S, what happens if you add a feature from x', and then what is the new output, if you take the expectation of all possible S'-s, that is the value of S.

(Refer Slide Time: 15:48)



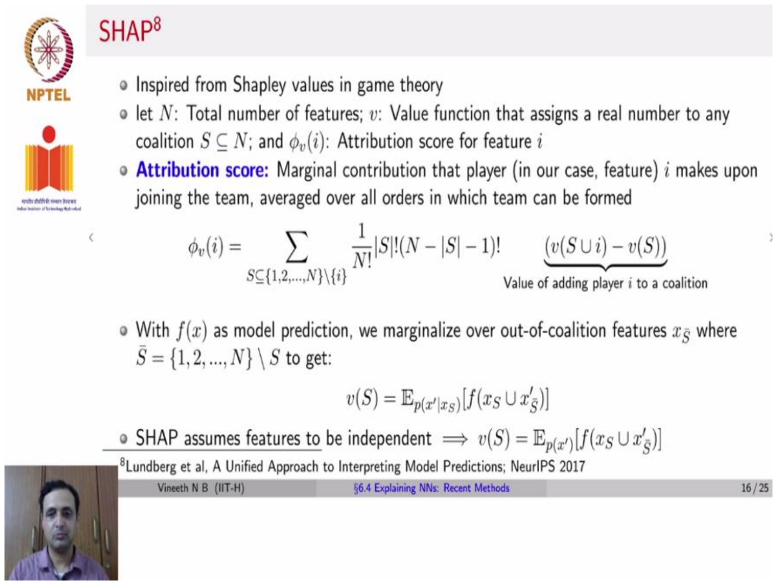
DeepSHAP

- Assumes input features are independent of one another and explanation model is linear
- Take distribution of baselines and compute DeepLIFT attribution for each input-baseline pair, then average resulting attributions per input example

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 17 / 25

There is an extension of SHAP called Deep SHAP, once again, where the input features are assumed to be independent of one another. And the explanation model is linear. But now, the reason it is called Deep SHAP, SHAP is it is a combination of SHAP and Deep Lift. And you take a distribution of baselines and compute the deep lift attribution for each input baseline pair. And then you average the resulting attributions per input example, to define your v or value function. The value function, in short, the value function is inspired by Deep lift. That is the idea here,

(Refer Slide Time: 16:31)



SHAP[8]

- Inspired from Shapley values in game theory
- let N: Total number of features; v: Value function that assigns a real number to any coalition $S \subseteq N$; and $\phi_v(i)$: Attribution score for feature i
- **Attribution score:** Marginal contribution that player (in our case, feature) i makes upon joining the team, averaged over all orders in which team can be formed

$$\phi_v(i) = \sum_{S \subseteq \{1,2,\ldots,N\} \setminus \{i\}} \frac{1}{N!} |S|!(N - |S| - 1)! \underbrace{(v(S \cup i) - v(S))}_{\text{Value of adding player } i \text{ to a coalition}}$$

- With $f(x)$ as model prediction, we marginalize over out-of-coalition features $x_{\bar{S}}$ where $\bar{S} = \{1, 2, \ldots, N\} \setminus S$ to get:

$$v(S) = \mathbb{E}_{p(x'|x_S)}[f(x_S \cup x'_{\bar{S}})]$$

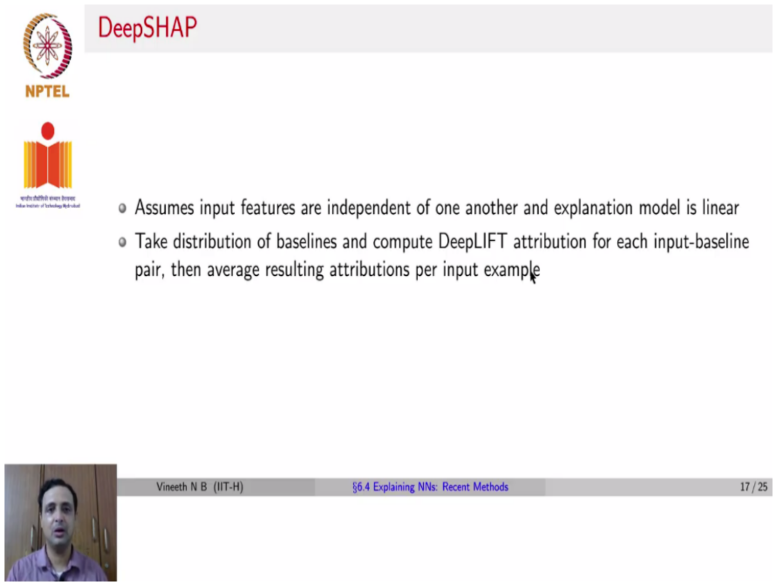
- SHAP assumes features to be independent $\implies v(S) = \mathbb{E}_{p(x')}[f(x_S \cup x'_{\bar{S}})]$

[8]Lundberg et al, A Unified Approach to Interpreting Model Predictions; NeurIPS 2017

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 16 / 25

The rest of it goes back and follows. So, the final attribution score would still be given by this equation here, we are only now talking about how the v is defined using these equations.

(Refer Slide Time: 16:45)



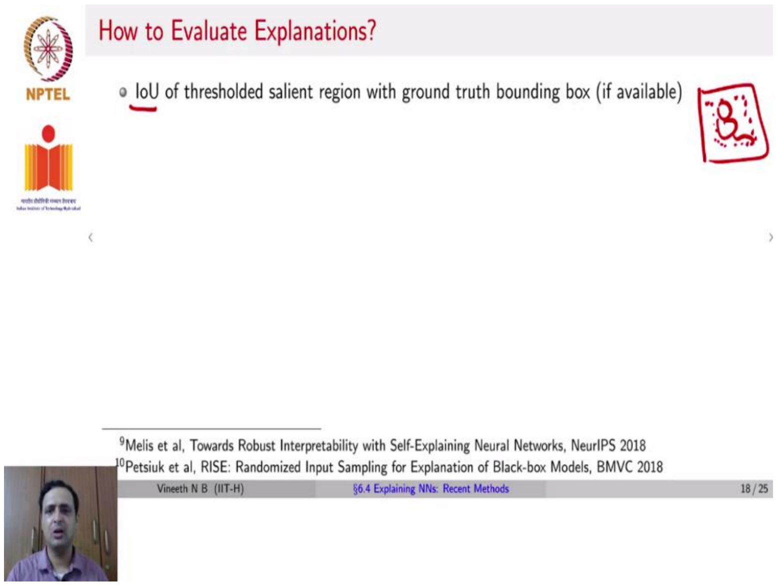
DeepSHAP

- Assumes input features are independent of one another and explanation model is linear
- Take distribution of baselines and compute DeepLIFT attribution for each input-baseline pair, then average resulting attributions per input example

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 17 / 25

Having seen these different methods to be able to explain the dishes, the decisions of CNNs or Neural networks, Neural network models.

(Refer Slide Time: 16:58)



How to Evaluate Explanations?

- IoU of thresholded salient region with ground truth bounding box (if available)

[9] Melis et al, Towards Robust Interpretability with Self-Explaining Neural Networks, NeurIPS 2018

[10] Petsiuk et al, RISE: Randomized Input Sampling for Explanation of Black-box Models, BMVC 2018

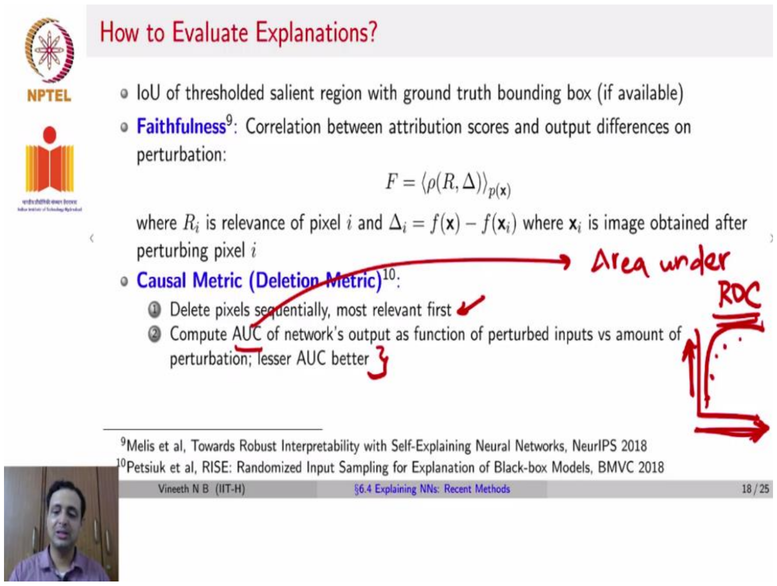
Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 18 / 25

An important question to ask is, how do you evaluate these explanations? Do you have an idea? How do you check the goodness of these explanations in all of the results that we showed so far, we showed qualitative results. So, we took an image and said, look at this, it seems to work well.

But when you have a million images in your test set, you really cannot look at every image in the test set and study whether CAM is working better, or Deep lift is working better or Deep SHAP is working better. You need some objective metric that averages the performance across all of your images. How do you do this? A simple way would be to take the intersection over the union IOU of the threshold salient region with a ground truth bounding box if it is available.

So, if somebody already told you that, given an image, I have a cat in the image, but I also am giving you a bounding box around the cat. Let us say that is part of the dataset, then if my saliency map has a high intersection over union with this particular cat, what is intersection over union, we will see this in more detail when we see other tasks such as Detection in vision. But if that intersection over union is high, we assume that our saliency map or our method to compute the saliency map has worked fairly well. What else can we use?

(Refer Slide Time: 18:30)



How to Evaluate Explanations?

- IoU of thresholded salient region with ground truth bounding box (if available)
- **Faithfulness**[9]: Correlation between attribution scores and output differences on perturbation:

$$F = \langle \rho(R, \Delta) \rangle_{p(\mathbf{x})}$$

where R_i is relevance of pixel i and $\Delta_i = f(\mathbf{x}) - f(\mathbf{x}_i)$ where $\mathbf{x}_i$ is image obtained after perturbing pixel i

- **Causal Metric (Deletion Metric)**[10]:
  1. Delete pixels sequentially, most relevant first
  2. Compute AUC of network's output as function of perturbed inputs vs amount of perturbation; lesser AUC better

[9] Melis et al, Towards Robust Interpretability with Self-Explaining Neural Networks, NeurIPS 2018
[10] Petsiuk et al, RISE: Randomized Input Sampling for Explanation of Black-box Models, BMVC 2018

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 18 / 25

There is another metric that is been recently proposed in NeurIPS 2018 and BMVC of 2018, which is called faithfulness. In faithfulness, we measure the correlation between the attribution scores and the output differences on perturbation. So, we are trying to see if if you perturb up input and you see an output change, then and if you see a significant output change, then the attribution of that input should also be high because that affected significant that affected the output significantly.

So, the correlation between the attribution score and the output differences on perturbation should ideally be high. So, here is the equation we say that faithfulness is given by rho which is the correlation of R comma delta R is the relevance of a particular pixel i and delta is the output difference when xi is the image obtained after perturbing pixel i. Another metric that was defined in the BMVC paper here is known as the causal metric.

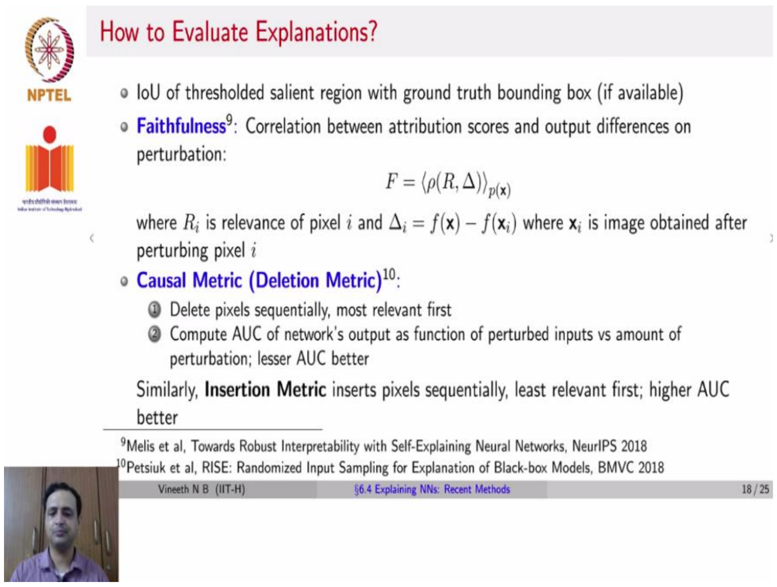
And it has two variants known as Deletion and Insertion metric. Let us see the deletion metric and the insertion metric turns out to be complimentary. So, the deletion metric you first delete seek pixels sequentially based on what our method says is most relevant first for a particular outcome. So, if we think the pixels of a cat is the most relevant for calling it a cat, remove those pixels from the image.

So, replace it with a black pixel or a gray pixel. So, remove those pixels from the image. And then you compute the AUC. I hope all of you know what AUC is. AUC is the area under the ROC curve, which is a standard metric is used in several machine learning settings. So, you compute your area under the ROC curve of the network's output, as so the area under the ROC curve, you would know has two axes.

Where generally you vary some threshold and see what happens to a certain set of metrics that you are monitoring. And you ideally want your ROC curve to be something like this with the high area, but in this particular case, we see what is the function of the perturbed inputs versus the amount of perturbation we are trying to see, as you keep increasing the perturbation? What happens to the function of perturbed inputs? Does it change a lot as you vary the x-axis, if the y-axis changes a lot, then perhaps those pixels do have a high influence on the output?

But because we have deleted the most relevant pixels, we would want the AUC to be lesser here, after the removal of a particular pixel. So if you remove the most relevant pixel, and plotted a ROC curve, the way we saw it here, which means you put up the input and see how much the output changes, and you plot this as a curve, you would want the AUC to be low after removing the most relevant pixel.

(Refer Slide Time: 21:55)



How to Evaluate Explanations?

- IoU of thresholded salient region with ground truth bounding box (if available)
- **Faithfulness**[9]: Correlation between attribution scores and output differences on perturbation:

$$F = \langle \rho(R, \Delta) \rangle_{p(\mathbf{x})}$$

where R_i is relevance of pixel i and $\Delta_i = f(\mathbf{x}) - f(\mathbf{x}_i)$ where $\mathbf{x}_i$ is image obtained after perturbing pixel i
- **Causal Metric (Deletion Metric)**[10]:
  1. Delete pixels sequentially, most relevant first
  2. Compute AUC of network's output as function of perturbed inputs vs amount of perturbation; lesser AUC better

Similarly, **Insertion Metric** inserts pixels sequentially, least relevant first; higher AUC better

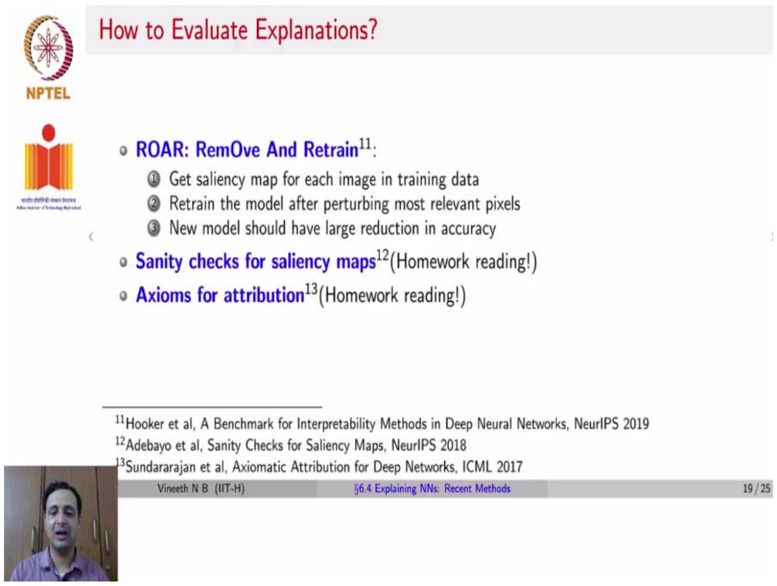
[9]Melis et al, Towards Robust Interpretability with Self-Explaining Neural Networks, NeurIPS 2018

[10]Petsiuk et al, RISE: Randomized Input Sampling for Explanation of Black-box Models, BMVC 2018

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 18 / 25

You could also look, look at this complementarily and see it as an insertion metric, where you insert pixels sequentially, least relevant first, in this case, a higher AUC is better. Just the way we spoke about it for the deletion metric.

(Refer Slide Time: 22:14)



How to Evaluate Explanations?

- **ROAR: RemOve And Retrain**[11]:
  1. Get saliency map for each image in training data
  2. Retrain the model after perturbing most relevant pixels
  3. New model should have large reduction in accuracy
- **Sanity checks for saliency maps**[12](Homework reading!)
- **Axioms for attribution**[13](Homework reading!)

[11]Hooker et al, A Benchmark for Interpretability Methods in Deep Neural Networks, NeurIPS 2019

[12]Adebayo et al, Sanity Checks for Saliency Maps, NeurIPS 2018

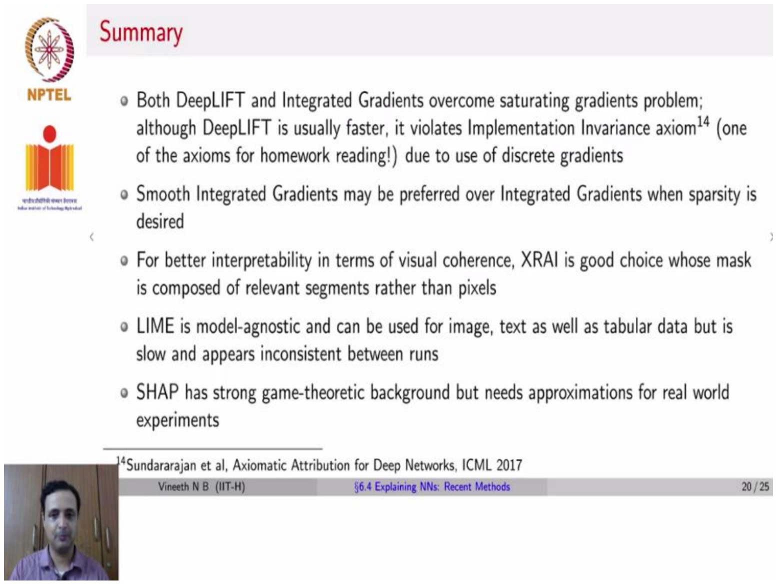
[13]Sundararajan et al, Axiomatic Attribution for Deep Networks, ICML 2017

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 19 / 25

Another recent metric that is been proposed in NeurIPS of 2019 is known as a Roar. Roar stands for \Remove and Retrain. What the Roar matrix suggests is you get a saliency map for each image in your training data. You retrain the model after perturbing the most relevant pixels, the new model should have a large reduction in accuracy. So if you take an image of a cat, you know what saliency map you got for a cat, you now perturb all of those pixels, you should ideally see a reduction in accuracy of the model calling it a cat.

That is the Roar metric. There have also been some things called Sanity checks for saliency maps that is been proposed, you are going to leave this as homework reading for you. And there is also a work that was published in ICML of 2017 known as Axiomatic Attributions, which defined a set of axioms for any of these attribution methods. It, it says that any attribution method is Grad CAM, be at Lime, be at Deep Lift, all the different methods that we saw this week must satisfy certain axioms for it to be for them to be reliable in practice. What are those axioms, once more homework for you to go back and read those axioms?

(Refer Slide Time: 23:40)



Summary

- Both DeepLIFT and Integrated Gradients overcome saturating gradients problem; although DeepLIFT is usually faster, it violates Implementation Invariance axiom[14] (one of the axioms for homework reading!) due to use of discrete gradients
- Smooth Integrated Gradients may be preferred over Integrated Gradients when sparsity is desired
- For better interpretability in terms of visual coherence, XRAI is good choice whose mask is composed of relevant segments rather than pixels
- LIME is model-agnostic and can be used for image, text as well as tabular data but is slow and appears inconsistent between runs
- SHAP has strong game-theoretic background but needs approximations for real world experiments

[14]Sundararajan et al, Axiomatic Attribution for Deep Networks, ICML 2017

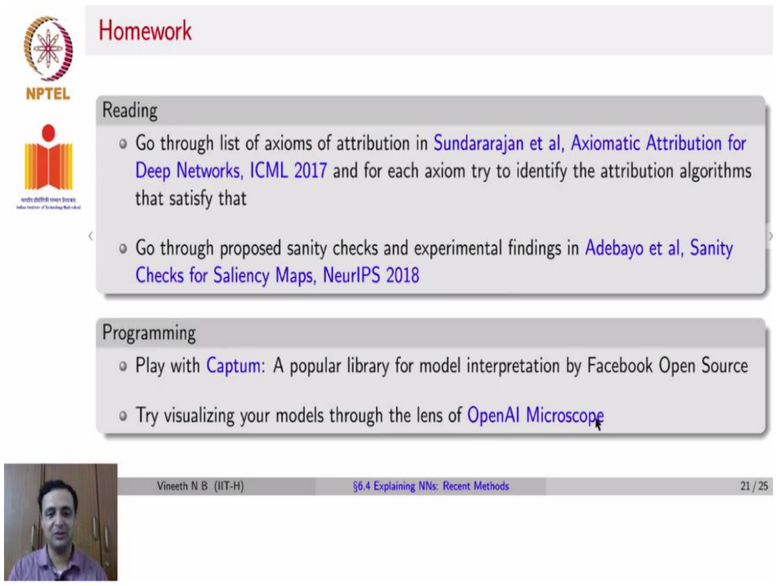
Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 20 / 25

To summarize this lecture, both Deep Lift and Integrated Gradients overcome the saturating gradients problem, although deep lift is generally faster because integrated gradients have to compute the data gradients across an entire path of different inputs. However, Deep lift violates one of the axioms we talked about on the previous slide, known as Implementation Invariance Axiom. Maybe you should try finding out why as homework.

Smooth integrated gradients are generally preferred over integrated gradients when sparsity is desired. For better interpretability in terms of visual coherence XRAI is a good choice, because it reasons at the level of regions rather than pixels. Lime is a model agnostic method. It can be used on top of any machine learning model because you can anyway good be approximated as a linear model at the end.

And it can be used for image data, text data, tabular data, so on and so forth. It is a bit slow because you have to train a model for every input image that you have. And is also sometimes inconsistent between runs. But it is a model that is used by many companies today. SHAP has a strong game-theoretic background. But to make it work, you have to use some approximations in the real world.

(Refer Slide Time: 25:12)



Homework

Reading

- Go through list of axioms of attribution in Sundararajan et al, Axiomatic Attribution for Deep Networks, ICML 2017 and for each axiom try to identify the attribution algorithms that satisfy that
- Go through proposed sanity checks and experimental findings in Adebayo et al, Sanity Checks for Saliency Maps, NeurIPS 2018

Programming

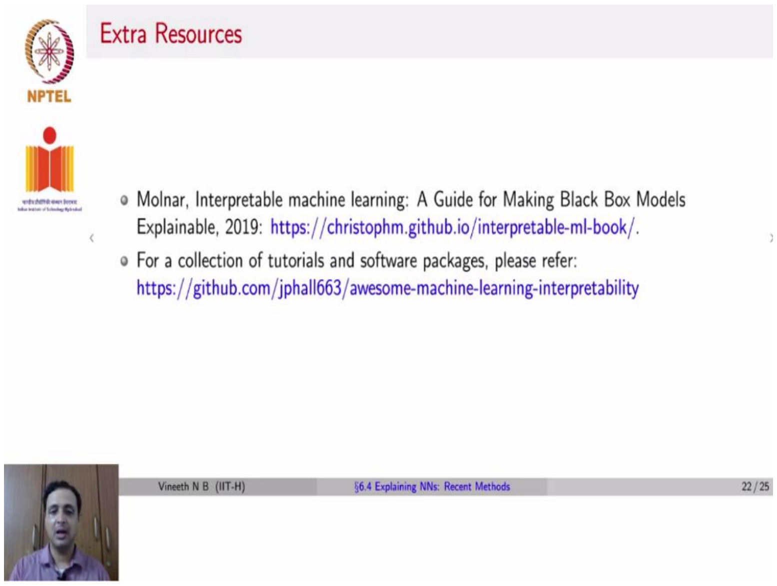
- Play with Captum: A popular library for model interpretation by Facebook Open Source
- Try visualizing your models through the lens of OpenAI Microscope

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 21 / 25

So, your homework is going to be about going through the list of axioms of attribution in this work. And for each axiom, try to find out the attribution methods that satisfy that the methods that we have covered this week that satisfy that particular axiom. Also, go through the proposed Sanity checks and experimental findings in this paper. Both of these papers are good reads.

And from a programming perspective, play with Captum it is a popular library for model interpretation by Facebook open source. You can also try visualizing any models that you have built-in your assignments so far, through the lens of OpenAI Microscope for visualization and understanding.

(Refer Slide Time: 25:57)



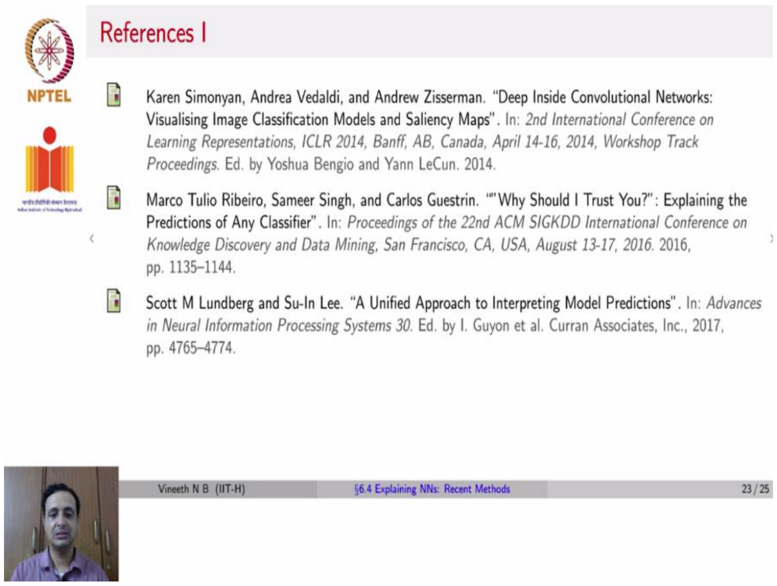
Extra Resources

- Molnar, Interpretable machine learning: A Guide for Making Black Box Models Explainable, 2019: https://christophm.github.io/interpretable-ml-book/.
- For a collection of tutorials and software packages, please refer: https://github.com/jphall663/awesome-machine-learning-interpretability

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 22 / 25

Here are some other resources. There is a very nice book by Christopher Molnar, known as Interpretable Machine Learning covers all of these methods in more detail if you are interested. It is an online book. There's also a collection of tutorials and software packages on this link that you see here.

(Refer Slide Time: 26:16)



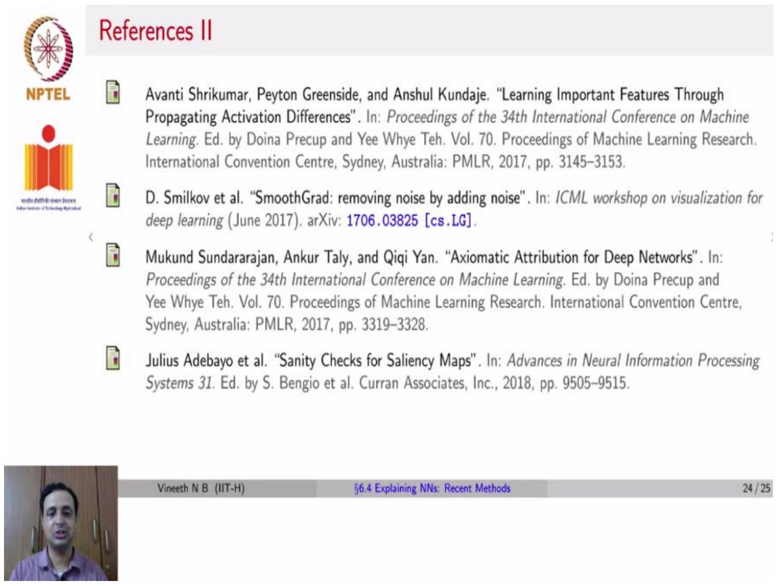
References I

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 1135–1144.

Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774.

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 23 / 25



References II

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 3145–3153.

D. Smilkov et al. "SmoothGrad: removing noise by adding noise". In: *ICML workshop on visualization for deep learning* (June 2017). arXiv: 1706.03825 [cs.LG].

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 3319–3328.

Julius Adebayo et al. "Sanity Checks for Saliency Maps". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 9505–9515.

Vineeth N B (IIT-H) §6.4 Explaining NNs: Recent Methods 24 / 25

The references for this lecture are here