**Two axioms of attribution methods:**

Sensitivity: This axiom states that an attribution method should be sensitive to the presence or absence of a feature. In other words, if a feature has a significant impact on the model's output for a particular input, the attribution method should assign a non-zero attribution score to that feature. Conversely, if a feature has no impact, its attribution score should be zero or close to zero.

Implementation Invariance: This axiom emphasizes that the attribution method's outcomes should be consistent and not dependent on the internal implementation details of the model. In other words, changing the model architecture, such as modifying the activation functions or retraining the model, should not drastically change the attribution results as long as the model's behavior (i.e., its predictions) remains consistent.

**Integrated Gradients:**

* Start with a baseline input where the feature is set to its neutral value.
* Gradually change the feature's value from the baseline to the actual input while recording the gradients.
* Integrate (accumulate) the gradients over this path to obtain an attribution score for the feature.

**Integrated Gradients:**

**Deeplift:(feature wise)**

Comparing the activations of each neuron in the network at a specific layer for a given input to the activations that would occur under a reference input (typically an input with all-zero values). It computes the differences between these two sets of activations.

Initialize Contribution Scores: Start with contribution scores initialized to zero for each input feature. These scores represent the attribution of the output to each feature.

Layer-by-Layer Calculation: DeepLIFT is applied layer by layer, starting from the output layer and moving backward through the network's layers.

For Each Layer:  
**a**. Compute the Rescale Factor: Calculate a rescale factor that quantifies how much the change in the output of the layer is due to changes in the input features. This factor helps ensure that the contribution scores sum up to the difference between the output of the actual input and a chosen reference input.  
**b**. Compute the Contribution Scores: For each neuron in the layer, compute the contribution scores for its input features. This is done by **taking the difference between the neuron's activation for the actual input and the reference input, scaling this difference by the rescale factor, and then multiplying by the gradient of the neuron's output with respect to its inputs.**  
**c**. Propagate Contributions Backward: Distribute the contribution scores computed in step 3b to the input features of the previous layer, taking into account the layer's activation function. This step ensures that contributions are properly attributed back through the network.

Normalization: After attributing contributions through all layers, it's common to perform additional normalization steps to ensure that the attribution scores are meaningful and add up to the difference between the output for the actual input and the reference input.

Result: The final attribution scores obtained after propagating through all layers provide insights into which input features had the most influence on the network's prediction.

**DeepLIFT is designed to satisfy the two axioms of attribution methods described in the paper "Axiomatic Attribution for Deep Networks" by Sundararajan et al. These axioms are:**

1. Sensitivity Axiom: This axiom states that if a feature's value is changed while keeping all other features fixed, the attribution score for that feature should change proportionally. In other words, the attribution method should be sensitive to changes in individual feature values.

DeepLIFT and Sensitivity Axiom: DeepLIFT satisfies the sensitivity axiom because it computes the importance of each feature by considering how changes in that feature affect the model's output relative to a reference input. It quantifies the sensitivity of the model's output to changes in individual features, aligning with the requirements of this axiom.

2. Implementation Invariance Axiom: This axiom requires that if two functionally equivalent networks are identical in every way, except for the implementation details (e.g., the choice of activation functions or the use of dropout), then the attribution scores should be the same for both networks.

DeepLIFT and Implementation Invariance Axiom: DeepLIFT aims to satisfy the implementation invariance axiom by considering the propagation of relevance scores (importance scores) through the network based on the network's operations. It attempts to produce consistent attribution results for functionally equivalent networks with different implementations. However, it's important to note that the extent to which this axiom is satisfied may depend on the specific implementation of DeepLIFT and the choice of rescaling rules.

While DeepLIFT strives to satisfy these axioms, it's worth noting that the degree to which attribution methods satisfy these axioms can vary based on the specific details of the implementation and the network architecture. Researchers and practitioners often use these axioms as guidelines for evaluating the effectiveness and fairness of attribution methods in practice.

**Grad Cam:(path based)**

* Grad-CAM (Gradient-weighted Class Activation Mapping) is a technique used in computer vision to visualize and interpret the decisions made by convolutional neural networks (CNNs), particularly in the context of image classification. Grad-CAM helps identify which parts of an image are most important in influencing the network's classification decision.
* Here's a step-by-step explanation of the Grad-CAM algorithm:
* 1. \*\*Input Image\*\*: Start with an input image that you want to interpret. This image is passed through the CNN, and its class label is predicted.
* 2. \*\*Forward Pass\*\*: Perform a forward pass of the image through the network until you get the class scores just before the final softmax layer. These scores represent the unnormalized probabilities for each class.
* 3. \*\*Backpropagation\*\*: Calculate the gradient of the score corresponding to the predicted class with respect to the feature maps of the last convolutional layer. This gradient highlights which neurons or feature maps were most responsible for the prediction.
* 4. \*\*Global Average Pooling (GAP)\*\*: Average the gradients obtained from the previous step for each feature map. This process produces a weight for each feature map that indicates its importance in making the final classification decision.
* 5. \*\*Weighted Sum of Feature Maps\*\*: Multiply each feature map by its corresponding weight and sum all the feature maps. This produces a weighted combination of the feature maps, which is a 2D heatmap.
* 6. \*\*ReLU Activation\*\*: Apply a ReLU activation to the heatmap to remove any negative values. This step highlights regions of the image that positively contributed to the predicted class.
* 7. \*\*Heatmap Visualization\*\*: Finally, resize the heatmap to the original image size, and overlay it on the original image to visualize which parts of the image were most important in making the classification decision.
* Here's an example of how Grad-CAM can be applied:
* Suppose you have a CNN model for image classification, and you want to interpret why it classified a particular image as a "cat."
* 1. Input Image: Use the image of the cat you want to interpret.
* 2. Forward Pass: Pass the cat image through the CNN model to obtain class scores.
* 3. Backpropagation: Calculate the gradient of the score corresponding to the "cat" class with respect to the feature maps of the last convolutional layer.
* 4. Global Average Pooling: Average the gradients obtained in step 3 for each feature map to get the importance weights for each feature map.
* 5. Weighted Sum of Feature Maps: Multiply each feature map by its corresponding weight and sum them to obtain a heatmap.
* 6. ReLU Activation: Apply ReLU to the heatmap to highlight important regions.
* 7. Heatmap Visualization: Overlay the heatmap onto the original cat image. The heatmap will indicate which regions of the image were most influential in classifying it as a cat.
* The resulting heatmap will show which parts of the image (e.g., the cat's face or tail) played a crucial role in the model's decision.
* Grad-CAM is a valuable tool for model interpretability and can help identify whether a network is focusing on the right regions of an image to make its classification decisions.

Doesn’t satisfy two axioms because, first, final visualization depends on the layer you choose to use, and different choices make the visualization different even though it’s the same model, so the second axiom is not satisfied. Second, it is designed specifically for cnn differentiable with a spatial resolution (of the chosen layer to use) that fits the scale of the original image

**Gradient\*Input**

Steps：

* Gradient Calculation: When applying the Gradient\*Input method, you start by computing the gradients of the model's output (usually the predicted class score) with respect to the input data. These gradients represent how sensitive the model's prediction is to changes in each individual pixel or feature of the input.
* Element-wise Multiplication: The obtained gradients are then element-wise multiplied with the original input data. This multiplication emphasizes the input features that have the most influence on the model's output. If a feature has a large positive gradient, it means that increasing that feature would increase the model's output, and vice versa.
* Visualization: The resulting element-wise product, which can be seen as a heatmap, is often normalized or post-processed to ensure that the values are interpretable. This heatmap is overlaid onto the original input image to highlight the regions or features that the model is most responsive to.

Interpretation:

* Regions in the heatmap with high positive values indicate features in the input data that strongly contribute to the model's decision.
* Regions with low or negative values suggest features that have a negative impact on the model's prediction.

**Guided Backpropagation**

* Backpropagation Basics:
  + Like traditional backpropagation, Guided Backpropagation starts with the model's final output and computes the gradient of the loss with respect to the input image.
* Guided ReLU Activation:
  + In traditional backpropagation, gradients can flow both in the positive and negative directions through the Rectified Linear Unit (ReLU) activation functions, which are commonly used in CNNs. This can make it challenging to discern which image features are important.
  + **Guided Backpropagation modifies the ReLU activation by allowing only positive gradients to propagate backward while setting negative gradients to zero. This means that only image regions that positively contributed to the model's output are emphasized, while regions with negative contributions are suppressed.**
* Propagation and Visualization:
  + The modified gradients, which only contain positive values, are propagated backward through the layers of the network. These gradients highlight the regions of the input image that are most relevant for the network's prediction.
  + These positive gradients can be visualized by overlaying them on the input image. The resulting visualization typically shows the regions where the model focused its attention when making a particular prediction.
* Interpretation:
  + Guided Backpropagation helps users understand which parts of the input image have the most positive impact on a specific model's decision. It highlights the features or patterns that are crucial for the prediction while ignoring unimportant or distracting information.

**Layer-wise Relevance Propagation (LRP)** is a technique used for interpreting deep learning models, particularly neural networks, by attributing the model's prediction to individual neurons and layers. LRP aims to explain how the model's input features contribute to its output in a transparent and understandable way. Here's a description of the method of Layer-wise Relevance Propagation when applied to interpret deep learning models:

* Forward Pass:
  + LRP begins with a forward pass through the neural network, where input data is processed layer by layer, leading to the final prediction. As each layer processes the data, it computes activations and transformations.
* Prediction Score:
  + The final prediction score or class probabilities are obtained at the output layer. This score is the result of the entire forward pass and represents the model's prediction for a given input.
* Backward Pass with Relevance:
  + The central idea of LRP is to perform a backward pass that distributes the prediction score (or relevance) backward through the layers of the network, attributing relevance to each neuron and input feature.
  + Initially, the prediction score is assigned as relevance to the output neurons, i.e., it is "back-propagated" to the output layer.
* Relevance Redistribution:
  + LRP redistributes relevance from the output layer back to the previous layer. This redistribution is guided by rules that ensure that relevance is fairly allocated to neurons in the previous layer based on their contribution to the output.
  + Different variants of LRP exist, such as LRP-$\alpha\beta$, which specifies rules for redistributing relevance. These rules aim to model the flow of information and importance within the neural network.
* Layer-Specific Rules:
  + LRP typically employs specific rules for each layer type in the network (e.g., fully connected layers, convolutional layers, and activation functions like ReLU). These rules are designed to capture the semantics of each layer's operation.
  + For example, in a convolutional layer, relevance may be distributed to the input pixels based on the convolutional filter weights and activation values.
* Normalization and Stabilization:
  + To ensure that the relevance scores are meaningful and do not explode or diminish during redistribution, LRP often includes normalization and stabilization steps.
* Interpretation and Visualization:
  + Once the relevance has been redistributed throughout the network, it can be visualized or analyzed to understand which input features and neurons were most relevant for the model's prediction.
  + Visualization often takes the form of heatmaps or saliency maps, where regions with higher relevance are emphasized, providing insights into which parts of the input were important for the prediction.

While this approach can be applied directly to generalized linear mappings,

product type non-linearities are not covered.

Other more traditional methods:

Calculate gradients of the output with respect to individual neurons or parameters to estimate their impact on the final prediction

Citation

[1]Zeiler, Matthew D. "Deconvolutional Networks." *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010.

Note: Deconvolutional Networks

[2]Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences." *International conference on machine learning*. PMLR, 2017.

Note: Deeplift

[3]Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." *Proceedings of the IEEE international conference on computer vision*. 2017.

Note: Grad-cam

[4]Shrikumar, Avanti, et al. "Not just a black box: Learning important features through propagating activation differences." *arXiv preprint arXiv:1605.01713* (2016).

Note: gradient\*input

[5]Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." *arXiv preprint arXiv:1412.6806* (2014).

Note: Guided backpropagation

[6]Binder, Alexander, et al. "Layer-wise relevance propagation for neural networks with local renormalization layers." *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25*. Springer International Publishing, 2016.

Note: Layerwise Relevance Propagation

[7]Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." *International conference on machine learning*. PMLR, 2017.

Note: Integrated Gradients

[8]Bau, David, et al. "Network dissection: Quantifying interpretability of deep visual representations." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.