Propagation-based explanations

CSEP 590B: Explainable AI

Ian Covert & Su-In Lee
University of Washington

Previously

- Feature importance explanations
  - Removal-based explanations
  - Shapley values

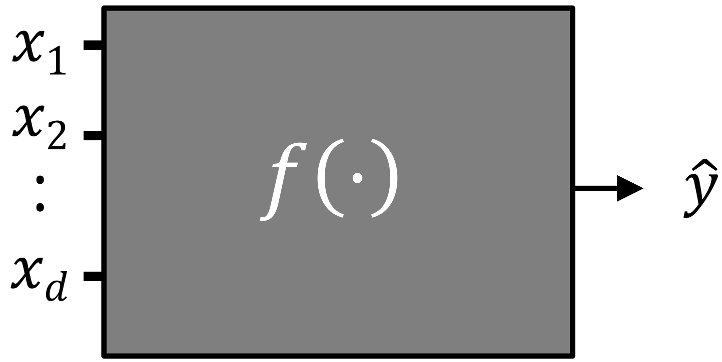
- Today: **propagation-based explanations**

Today

- Section 1
  - Backprop review
  - Gradient-based explanations
- Section 2
  - Modified backprop variants
  - Propagation vs. removal-based explanations

Setup

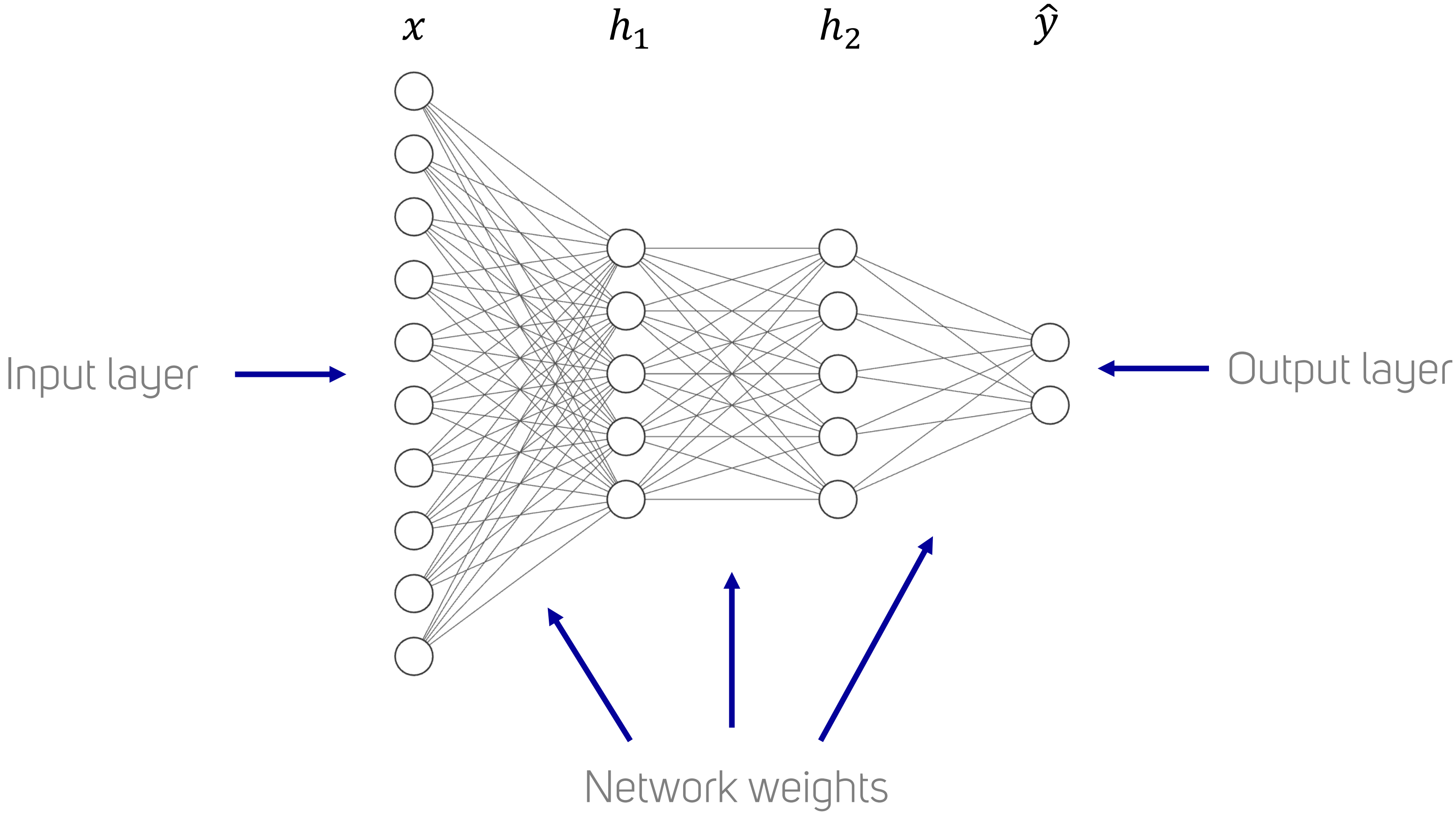
- Consider a classification model $f(x)$
- $f_y(x)$ is probability for class y
- Input is $x \in \mathbb{R}^d$ (must be continuous)
- Assume f is differentiable (a neural network)



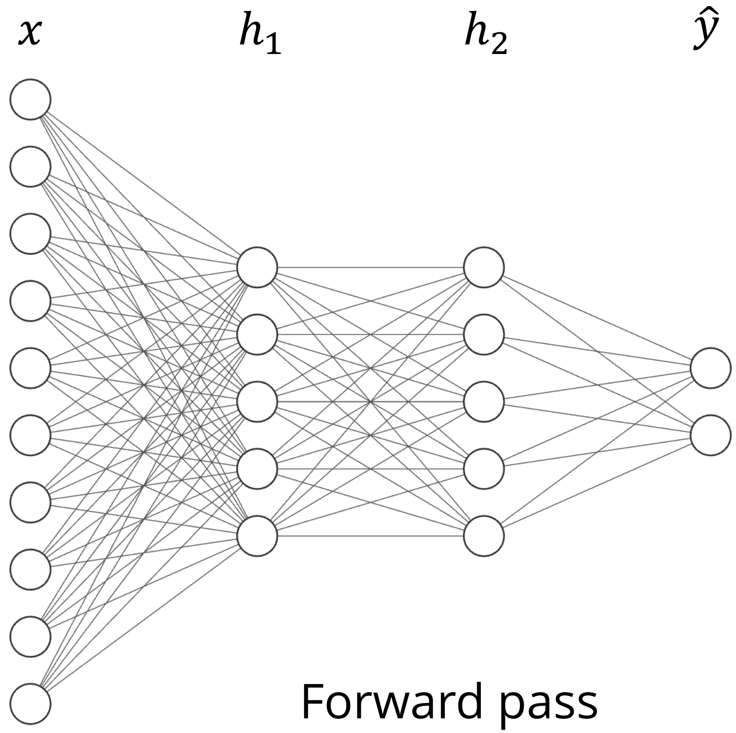
Review: backpropagation

- Deep learning models are trained using stochastic gradient descent (SGD)
  - Get a minibatch of examples
  - Calculate predictions and loss
  - Calculate gradients using “backprop” algorithm

Backpropagation



Backpropagation



Gradient descent

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

Network parameters

$$\theta = ([w_1, b_1], [w_2, b_2], [w_{\text{out}}, b_{\text{out}}])$$

Forward pass

$$h_1 = \sigma(w_1 x + b_1)$$
$$h_2 = \sigma(w_2 h_1 + b_2)$$
$$\hat{y} = w_{\text{out}} h_2 + b_{\text{out}}$$

Backpropagation (cont.)

- Model loss is mean prediction error:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(f\left(x^i;\theta\right), y^i\right)$$

- Gradient calculation:

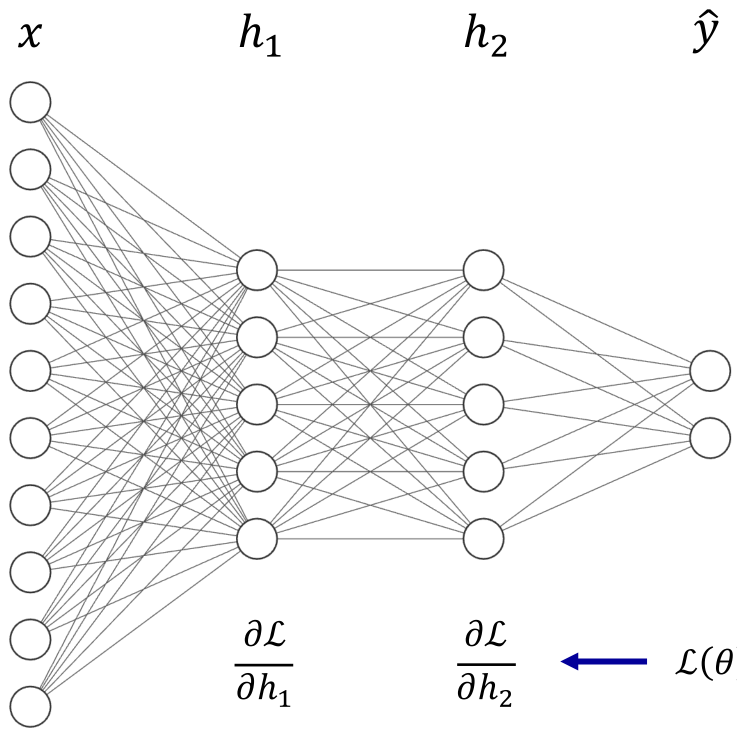
$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ell\left(f\left(x^i;\theta\right), y^i\right)$$

Chain rule

- Calculate gradients for all parameters θ using chain rule
  - Get gradients for last hidden layer
  - Then for the previous layer
  - Then the layer before that…
- Backpropagation = propagating gradients backward through the network

Rumelhart et al., “Learning representations by back-propagating errors” (1986)

Chain rule (cont.)



$$\frac{\partial \mathcal{L}}{\partial h_1} \qquad \frac{\partial \mathcal{L}}{\partial h_2} \longleftarrow \mathcal{L}(\theta) = \sum \ell(\hat{y}, y)$$

$$\frac{\partial \mathcal{L}}{\partial w_1} \longleftarrow \frac{\partial \mathcal{L}}{\partial w_2} \longleftarrow \frac{\partial \mathcal{L}}{\partial w_{\text{out}}}$$

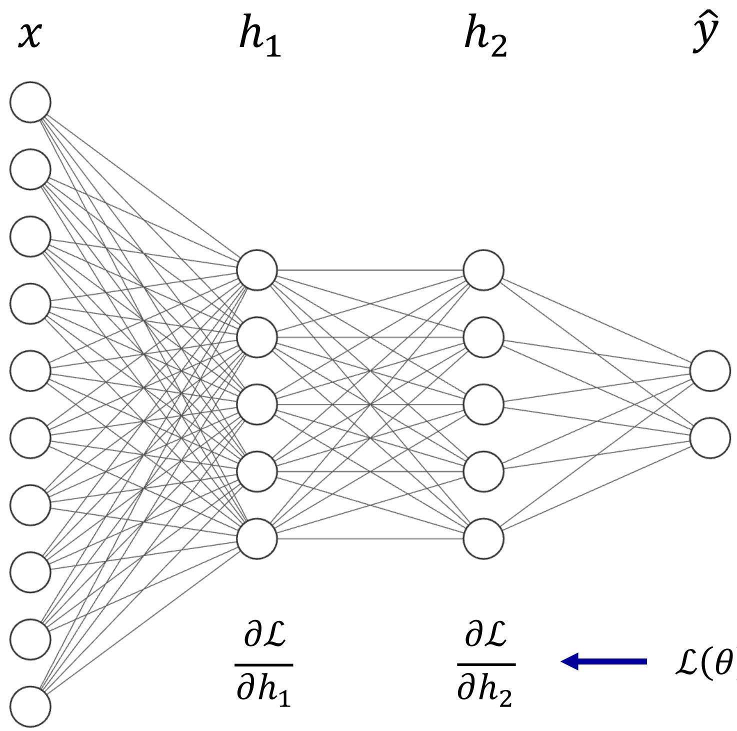
$$\frac{\partial \mathcal{L}}{\partial b_1} \qquad \frac{\partial \mathcal{L}}{\partial b_2} \qquad \frac{\partial \mathcal{L}}{\partial b_{\text{out}}}$$

Propagation-based explanations

- Use backprop idea to quantify feature importance

- Rather than gradients w.r.t. **parameters**, calculate gradients w.r.t. **inputs**

Input gradients

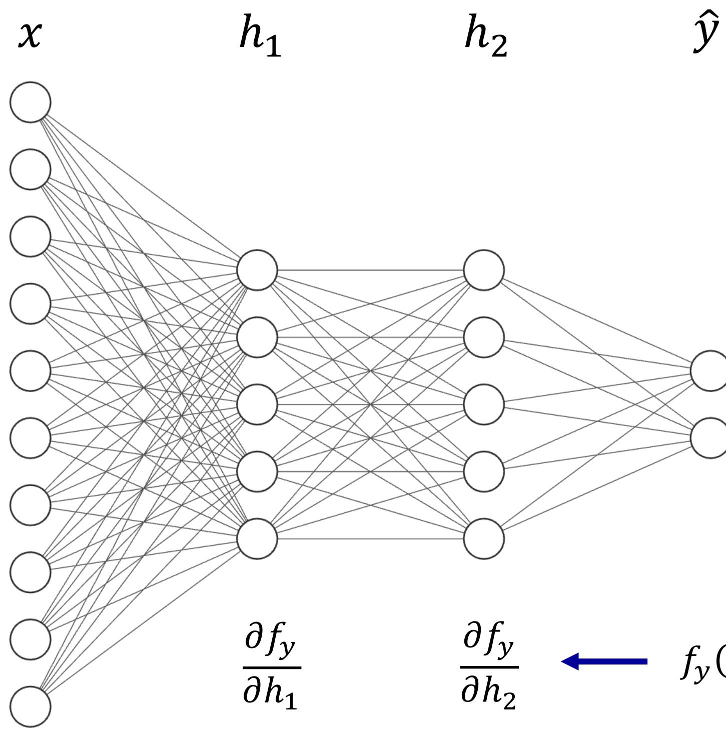


$$\frac{\partial \mathcal{L}}{\partial h_1} \qquad \frac{\partial \mathcal{L}}{\partial h_2} \leftarrow \mathcal{L}(\theta) = \sum \ell(\hat{y}, y)$$

$$\frac{\partial \mathcal{L}}{\partial x} \leftarrow \frac{\partial \mathcal{L}}{\partial w_1} \leftarrow \frac{\partial \mathcal{L}}{\partial w_2} \leftarrow \frac{\partial \mathcal{L}}{\partial w_{\text{out}}}$$

$$\frac{\partial \mathcal{L}}{\partial b_1} \qquad \frac{\partial \mathcal{L}}{\partial b_2} \qquad \frac{\partial \mathcal{L}}{\partial b_{\text{out}}}$$

Input gradients



$$\frac{\partial f_y}{\partial h_1} \qquad \frac{\partial f_y}{\partial h_2} \qquad f_y(x;\theta) = \hat{y}$$

Gradient of prediction (instead of loss)

$$\frac{\partial f_y}{\partial x} \qquad \frac{\partial f_y}{\partial w_1} \qquad \frac{\partial f_y}{\partial w_2} \qquad \frac{\partial f_y}{\partial w_{\text{out}}}$$

w.r.t. model input

$$\frac{\partial f_y}{\partial b_1} \qquad \frac{\partial f_y}{\partial b_2} \qquad \frac{\partial f_y}{\partial b_{\text{out}}}$$

Intuition

- Partial derivatives represent sensitivity to small perturbations

- Mathematically:

Delta from small change in ith direction

$$\frac{\partial f_y}{\partial x_i}(x) = \lim_{\epsilon \to 0} \frac{f_y(x + e_i \cdot \epsilon) - f_y(x)}{\epsilon}$$

Limit as change becomes very small

Measured relative to the size of change

Today

- Section 1
  - Backprop review
  - Gradient-based explanations
- Section 2
  - Modified backprop variants
  - Propagation vs. removal-based explanations

Application to XAI

- **Idea:** find features that cause large output changes when perturbed

- **Remark:** quantifies feature sensitivity, but not necessarily related to feature removal

Vanilla gradients

- For an input x and label y, calculate gradient of the prediction $f_y(x)$:

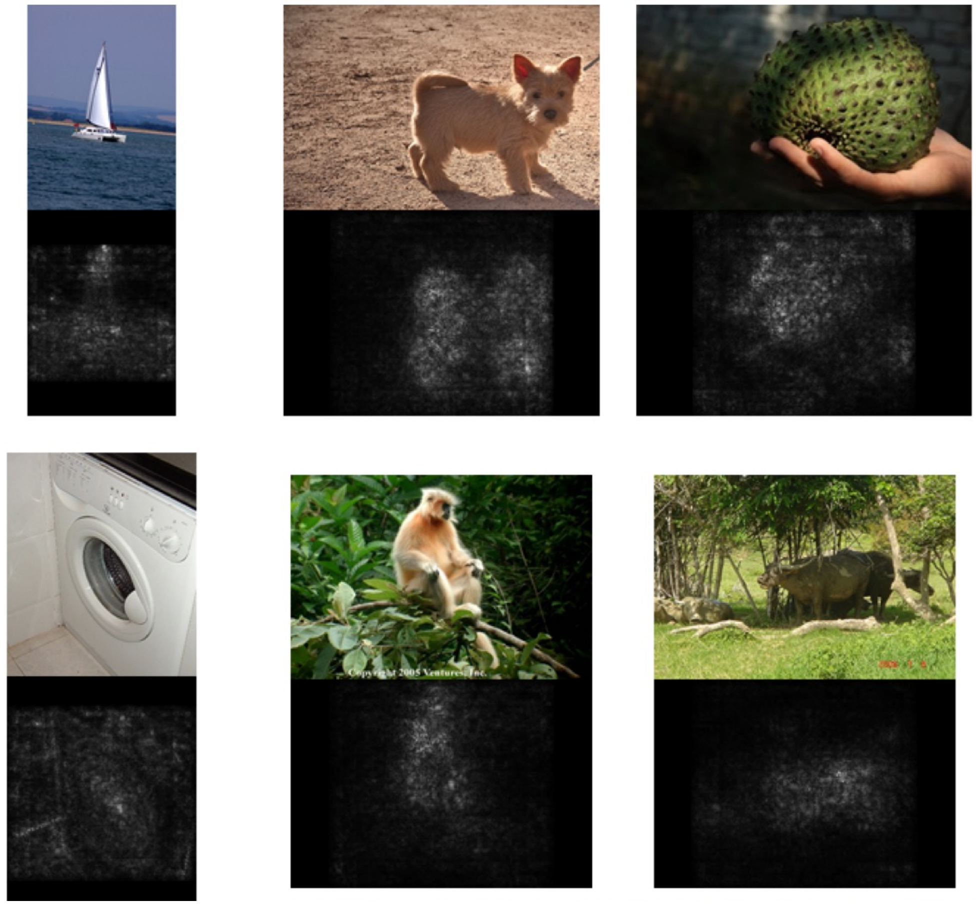
$$a_i = \frac{\partial f_y}{\partial x_i}(x)$$

- Can optionally use absolute value:

$$a_i = \left| \frac{\partial f_y}{\partial x_i}(x) \right|$$

Simonyan et al., "Deep inside convolutional networks: Visualizing image classification models and saliency maps" (2014)

Vanilla gradients (cont.)



Variation 1: SmoothGrad

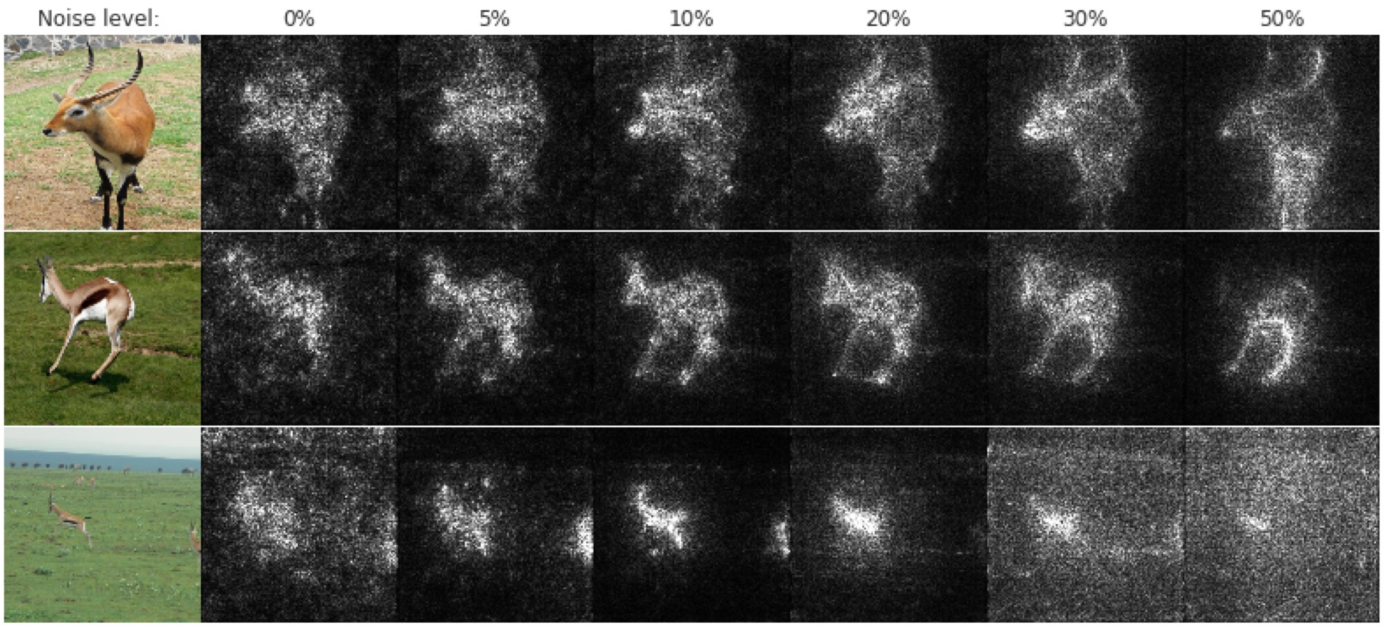
- Average gradients across inputs near x
- E.g., add Gaussian noise:

$$a_i = \mathbb{E}\left[\frac{\partial f_y}{\partial x_i}(x+\epsilon)\right] \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

- In practice, use small number of ϵ samples (50)
- Must tune σ^2 to an appropriate level

Smilkov et al., "SmoothGrad: Removing noise by adding noise" (2017)

SmoothGrad (cont.)



Standard saliency maps

Varying levels of input noise

Variation 2: Grad x Input

- Multiply gradient by input values:

$$a_i = x_i \cdot \frac{\partial f_y}{\partial x_i}(x)$$

Shrikumar et al., "Not just a black box: Learning important features through propagating activation differences" (2016)

Grad x Input (cont.)

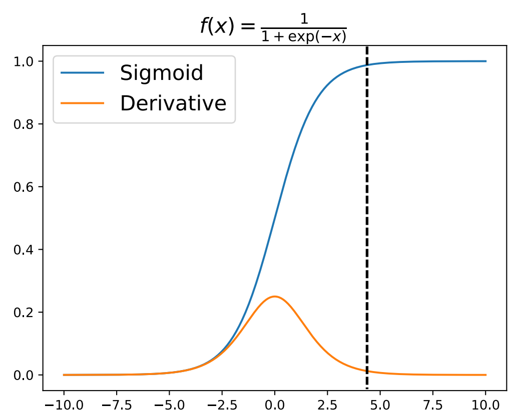
- **Interpretation:** consider the model's first-order Taylor expansion around x_0

$$f_y(x) \approx f_y(x_0) + (x - x_0)^\top \frac{\partial f_y}{\partial x}(x_0)$$

- Gradient gives linearized version of model (like replacing a function with its tangent line)
- Grad x Input = approximates impact of setting the input to zero
  - Similar to occlusion (see previous lecture)

Variation 3: Integrated gradients

- Gradients can become “saturated”
- Model is sensitive to big input changes, but not small ones



In this region, $f(x)$ is insensitive to small x changes

Gradient ≈ 0

- Saturation can yield small gradients, even for important inputs

Sundararajan et al., “Axiomatic attribution for deep networks” (2017)

IntGrad (cont.)

- **Idea:** address saturation issue by calculating gradients for rescaled images, $\alpha \cdot x$

$$\frac{\partial f_y}{\partial x_i}(\alpha \cdot x) \text{ for } 0 \leq \alpha \leq 1$$

- Integrate (average) gradients across range of rescaled images:

$$\int_{\alpha=0}^{1} \frac{\partial f_y}{\partial x_i}(\alpha \cdot x)\, d\alpha$$

- Multiply by the input feature value:

$$a_i = x_i \int_{\alpha=0}^{1} \frac{\partial f_y}{\partial x_i}(\alpha \cdot x)\, d\alpha$$

IntGrad (cont.)

- Implicitly relies on a zeros baseline
- Can instead use a non-zero baseline x'

$$a_i = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\partial f_y}{\partial x_i} \big(x' + \alpha \cdot (x - x')\big)\, d\alpha$$

- Related to a different idea from cooperative game theory: the Aumann-Shapley value
  - Different from previous Shapley value
  - Has its own axiomatic derivation (see Sundararajan et al.)

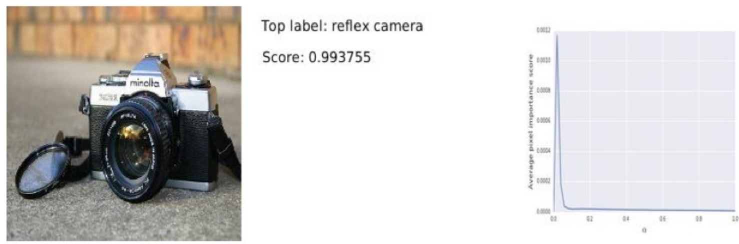
IntGrad (cont.)

- **Problem:** the integral is hard to calculate

- **Solution:** use Riemann sum approximation for m regularly spaced values $\alpha_j \in [0,1]$:

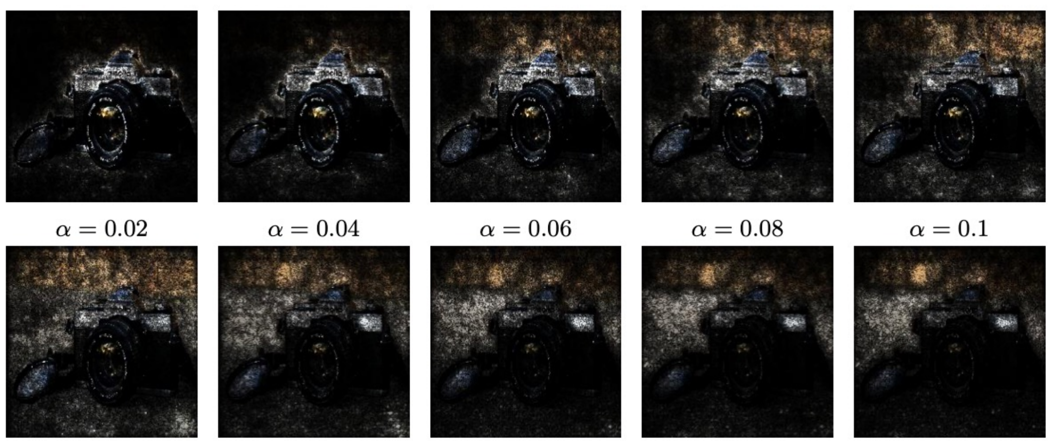
$$a_i \approx (x_i - x_i') \frac{1}{m} \sum_{j=1}^{m} \frac{\partial f_y}{\partial x_i}\left(x' + \alpha_j \cdot (x - x')\right)$$

IntGrad (cont.)

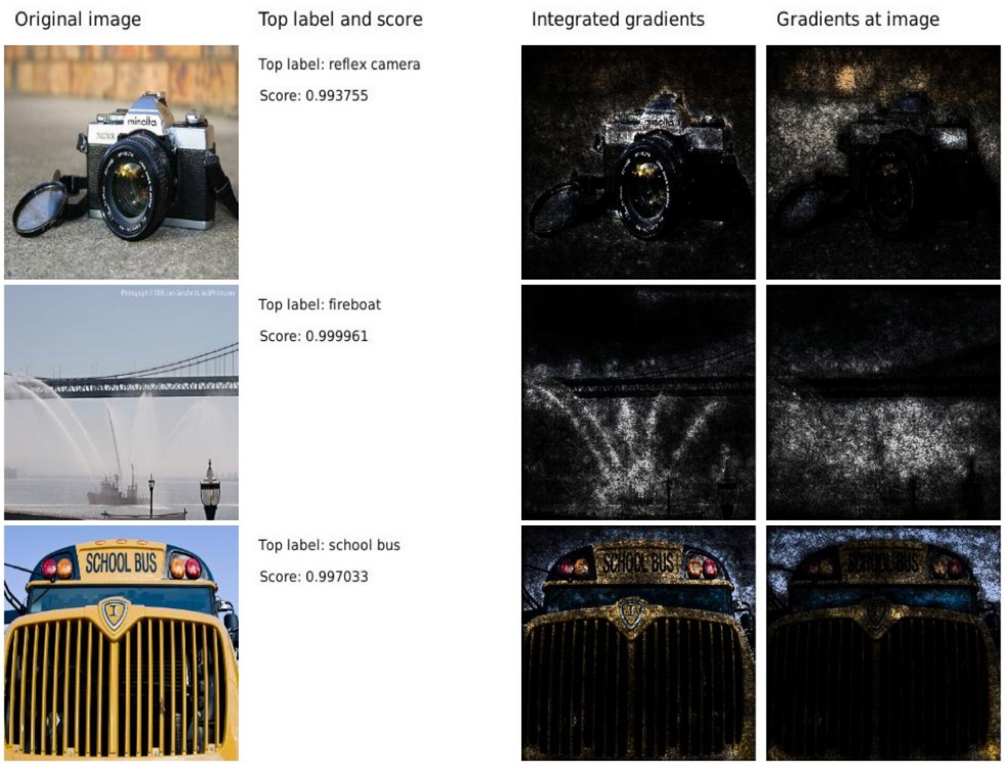


Input image and trend of the pixel importance scores obtained from interior gradients.

Gradient with image rescaled by α



IntGrad (cont.)



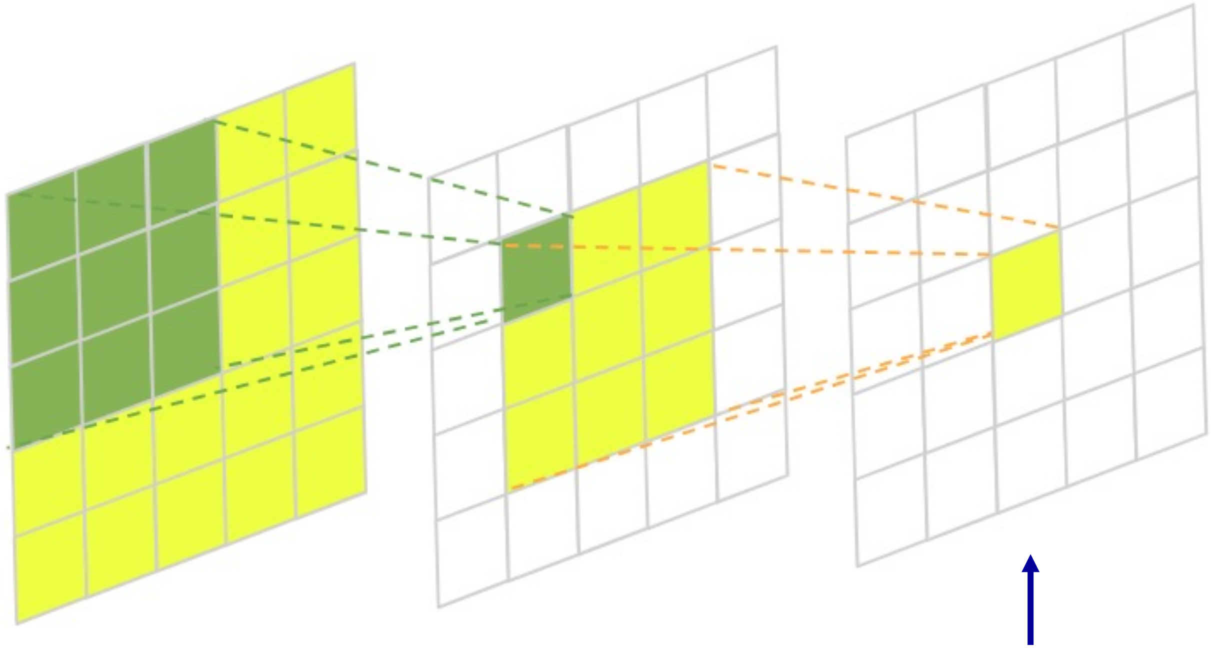
GradCAM

- In CNNs, hidden layers represent high-level visual concepts
- Hidden layers retain spatial information due to convolutional structure

- **Idea:** explain models via the last convolutional layer instead of the input layer

Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization" (2017)

CNN receptive fields



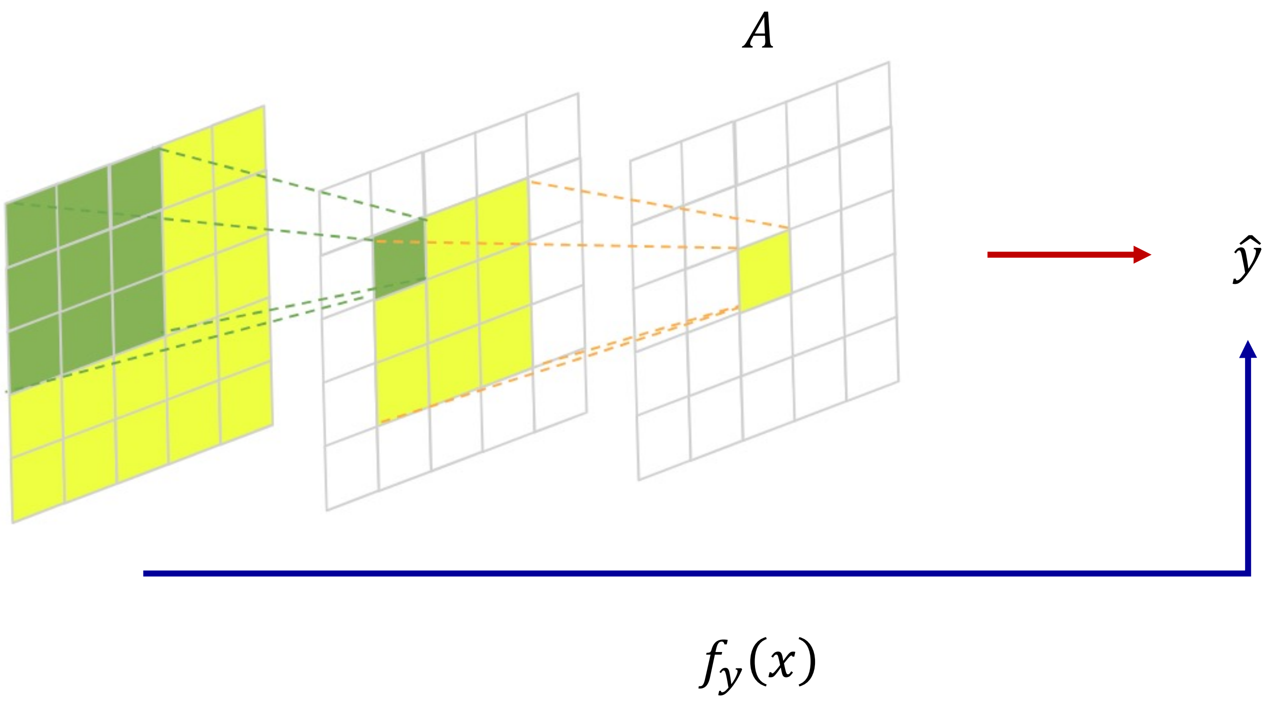
Receptive field grows at each layer, but remains localized

GradCAM procedure

- Denote final layer's hidden representation as A
  - Size is $A \in \mathbb{R}^{w \times h \times c}$
  - Width w, height h, channels c
  - Each channel $k = 1, \dots, c$ denoted as $A^k \in \mathbb{R}^{w \times h}$

- The final prediction $f_y(x)$ can be viewed as a function of representation A
  - E.g., $A \rightarrow$ global average pooling $\rightarrow$ MLP

GradCAM procedure (cont.)



GradCAM procedure (cont.)

- Calculate gradients w.r.t. A

$$\frac{\partial f_y}{A_{ij}^k} \quad \text{for all } (i, j, k)$$

- Average gradients within each channel:

$$\alpha_k^y = \frac{1}{wh} \sum_{ij} \frac{\partial f_y}{A_{ij}^k}$$

- Aggregate hidden representations across channel dimension using α_k^y

$$a_{ij} = \sum\nolimits_{k=1}^{c} \alpha_k^y A_{ij}^k$$

GradCAM procedure (cont.)

- Often use thresholding function (suppress negative attributions):

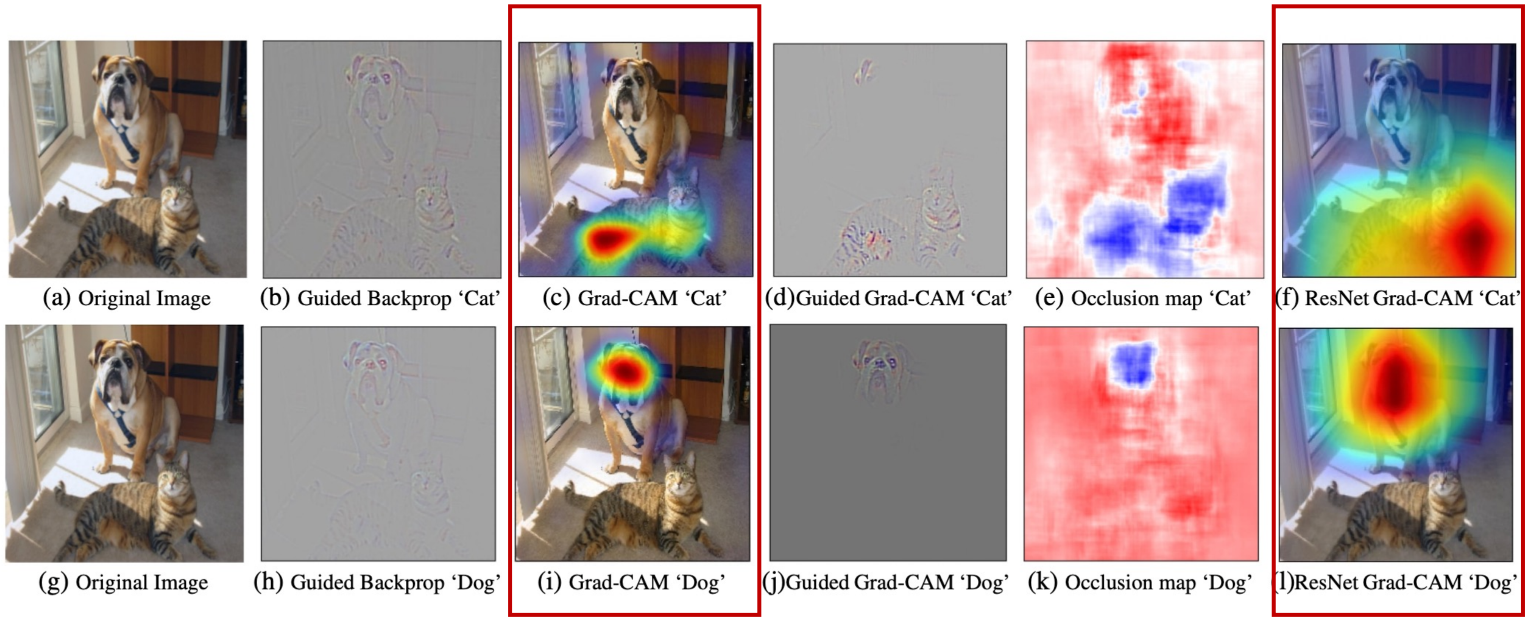
$$a_{ij} = \text{ReLU}\left(\sum_{k=1}^{c} \alpha_k^y A_{ij}^k\right)$$

- Can optionally upsample low-resolution scores a_{ij} to the original input size (e.g., bilinear upsampling)

GradCAM interpretation

- The values α_k^y represent smoothed or averaged gradient of class y w.r.t. channel k
- At each location, activations A_{ij}^k are multiplied by averaged gradients and then aggregated
- Similar to Grad x Input, but using a hidden layer instead of input layer
  - Like a Taylor approximation of setting internal activations to zero

GradCAM results



Other gradient-based methods

- **Guided backprop:** Springenberg et al., “Striving for simplicity: The all-convolutional net” (2014)
- **VarGrad:** Adebayo et al., “Local explanation methods for deep neural networks lack sensitivity to parameter values” (2018)
- **Expected gradients:** Erion et al., “Learning explainable models using attribution priors” (2020)
- **BlurIG:** Xu et al., “Attribution in scale and space” (2020)

Today

- Section 1
  - Backprop review
  - Gradient-based explanations
  - **10 min break**
- Section 2
  - Modified backprop variants
  - Propagation vs. removal-based explanations

Propagation-based explanations (continued)

CSEP 590B: Explainable AI

Ian Covert & Su-In Lee
University of Washington

Today

- Section 1
  - Backprop review
  - Gradient-based explanations
- Section 2
  - Modified backprop variants
  - Propagation vs. removal-based explanations

Modified backpropagation

- Previous approaches rely on **gradient** backprop
- Others use heuristic backprop variants
  - Calculate "importance" of internal nodes, propagate back to earlier ones
  - Requires justification for different backprop heuristics

Layer-wise relevance propagation (LRP)

- Intuition: iteratively calculate relevance scores for every layer of a model
  - Start with nodes in the last hidden layer
  - Move backwards through the model by splitting scores in the previous layer

Bach et al., "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation" (2015)

LRP (cont.)

- Let $R_j^{(l+1)} \in \mathbb{R}$ denote relevance for jth node in layer $l+1$
- Initialize $R_1^{(L)} = f_y(x)$ for output node of interest
- Let $R_{i\leftarrow j}^{(l,l+1)}$ denote *relevance message* sent from jth node in layer $l+1$ to ith node in layer l
- Want messages to satisfy two *conservation properties*

$$R_j^{(l+1)} = \sum_i R_{i\leftarrow j}^{(l,l+1)}$$

Summation of outgoing importance

Summation of incoming importance

$$R_i^{(l)} = \sum_j R_{i\leftarrow j}^{(l,l+1)}$$

LRP (cont.)

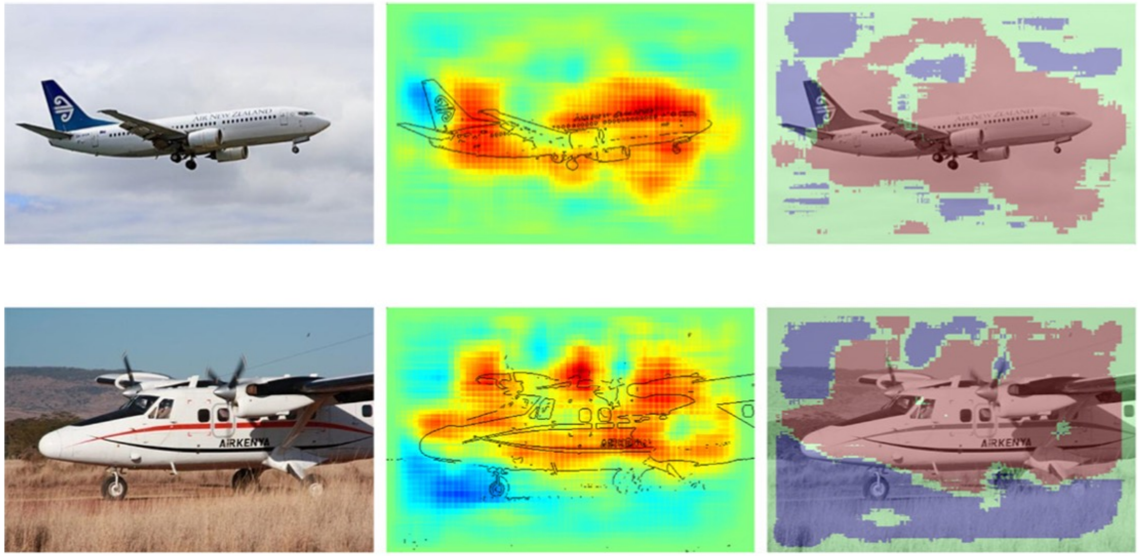
- The previous rules don't define a unique procedure, so the authors propose multiple options
- For example, the "ϵ-rule"

$$R_{i \leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j + \epsilon \cdot \mathrm{sign}(z_j)} R_j^{(l+1)}$$

where $z_{ij} = w_{ij}^{(l+1)} h_i^{(l)}$, $z_j = \sum_i z_{ij} + b_j^{(l+1)}$, and $\epsilon > 0$

- Finally, attributions are given by $a_i = R_i^{(1)}$ for $i = 1, \dots, d$

LRP (cont.)



LRP discussion

- Arguably less intuitive than other methods, requires some heuristic choices (which “rule” to use?)
- Can be difficult to adapt to different architectures
  - E.g., does not automatically support residual connections (ResNet architecture), requires extension to transformers

Other modified backpropagation methods

- **DeepLIFT:** Shrikumar et al., "Not just a black box: Learning important features through propagating activation differences" (2016)
- **PatternAttribution:** Kindermans et al., "Learning how to explain neural networks: PatternNet and PatternAttribution" (2017)
- **A unifying perspective:** Ancona et al., "Towards better understanding of gradient-based attribution methods" (2017)
- **Excitation backprop:** Zhang et al., " Top-down neural attention by excitation backprop" (2018)
- **LRP variants:** Montavon et al., "Layer-wise relevance propagation: an overview" (2019)

Today

- Section 1
  - Backprop review
  - Gradient-based explanations
- Section 2
  - Modified backprop variants
  - Propagation vs. removal-based explanations

Many explanation methods

- Removal-based explanations
  - SHAP, LIME, RISE, Occlusion, permutation tests

- Propagation-based explanations
  - SmoothGrad, IntGrad, GradCAM

What should we use in practice?

Model flexibility

What kind of model are you explaining?

- Removal-based explanations are **model-agnostic**
  - Can work with any model class (DNNs, trees, etc.)
- Propagation-based explanations are mainly for neural networks
  - Usually require differentiation
  - Some even have architecture constraints

Data flexibility

What kind of data do you have?

- Removal-based explanations can handle discrete and continuous features
  - E.g., replace inputs with alternative values from the dataset

- Propagation-based methods only make sense for **continuous** features
  - Derivative = sensitivity to small change
  - Small changes are meaningless for discrete features
    - E.g., $x_i \in \{0, 1, 2\}$

Local or global

What kind of explanation do you need?

- Both removal- and propagation-based methods can produce local explanations

- Removal-based methods are better suited for global explanations
  - Can focus on a global model behavior (e.g., dataset loss)
  - To use a propagation-based method, we require an aggregation scheme (e.g., mean of local explanations)

Speed

Is speed important?

- Propagation-based methods are **fast**
  - Backward pass through DNN
  - Weak dependence on number of features

- Removal-based methods can be **slow**
  - Often require making predictions with many feature subsets
  - Shapley values are particularly challenging

Quality

Which explanation is most informative or correct?

- Theory can serve as a guide
  - E.g., Shapley value axioms, IntGrad axioms
- We can also take an empirical approach
  - Metrics for explanation quality (next lecture)

- **Perspective:** no explanation is *wrong*, but some procedures are misaligned with user questions

Popular methods

Which methods do most people use?

- A small number of methods dominate
- Depends on the data domain (tabular, image, NLP)

Tabular data

- **Permutation tests** are widely used for global feature importance

- **SHAP** is ubiquitous for local explanations
  - TreeSHAP is built into XGBoost, LGBM
  - KernelSHAP used for other models

Computer vision

- Gradient-based methods are currently most popular: **GradCAM**, **IntGrad**
  - Removal-based methods are usually too slow
  - Some papers try to fix this, but not popular (yet?)
    - Masking model: Dabkowski & Gal, "Real time image saliency for black box classifiers" (2017)
    - CXPlain: Schwab & Karlen, "CXPlain: Causal explanations for model interpretation under uncertainty" (2019)
    - FastSHAP: Jethani et al., "FastSHAP: Real-time Shapley value estimation" (2021)

NLP

- NLP models (LSTMs, transformers) can use most methods
  - Gradient-based methods are popular
  - Removal-based explanations are slower, but leave-one-out (occlusion) is sometimes used

- For transformers, some use **attention** as explanation
  - Perhaps an *interpretable architecture*?
  - We'll return to this in a later lecture

Popular packages

GitHub package	Description	Stars
slundberg/shap	SHAP variations (KernelSHAP, TreeSHAP, DeepSHAP, etc.)	15.8k
marcotcr/lime	LIME for images, tabular data	9.7k
utkuozbulak/pytorch-cnn-visualizations	Various gradient-based methods	6.4k
jacobgil/pytorch-grad-cam	GradCAM + GradCAM variations	4.5k
pytorch/captum	Various gradient-based methods + SHAP	3.0k
sicara/tf-explain	Various gradient-based methods + occlusion	906
kundajelab/deeplift	DeepLIFT	616
ankurtaly/Integrated-Gradients	IntGrad	453