University of Idaho

CS 502
Directed Studies: Adversarial Machine Learning

Dr. Alex Vakanski

Lecture 14

Explainability in Machine Learning

Lecture Outline

- Explainability in ML
  - Concerns by stakeholders and for deployment
  - Transparency
  - Transparent versus opaque models
- Type of explanations
  - Pixel-level explanations
    - Vanilla BackProp, Guided BackProp, Occlusion maps, CAM, Grad-CAM, Guided Grad-CAM, Integrated gradients, Layer-wise relevancy propagation
  - Feature-level explanations
    - LIME, Shapley values, SHAP
  - Concept-level explanations
    - TCAV, ACE
  - Instance-level explanations
    - Prototypes and criticisms, counterfactual explanations

Explainability in ML

- Several terms are used interchangeably for ***explainability in ML***
  - Explainable ML
  - Explainable AI (with the acronym XAI)
  - Interpretable ML, or interpretable AI
- ML systems are increasingly being deployed across a wide range of applications
  - Most studies focus on increasing the predictive accuracy of ML models
  - Beside achieving high accuracy, we need to also have good understanding of the internal working of ML models
  - Current best performing models (DL-based models) are the least transparent
- Challenges:
  - Do we understand the decisions suggested by ML models?
  - Can we trust ML models if their decision-making process is not fully transparent?

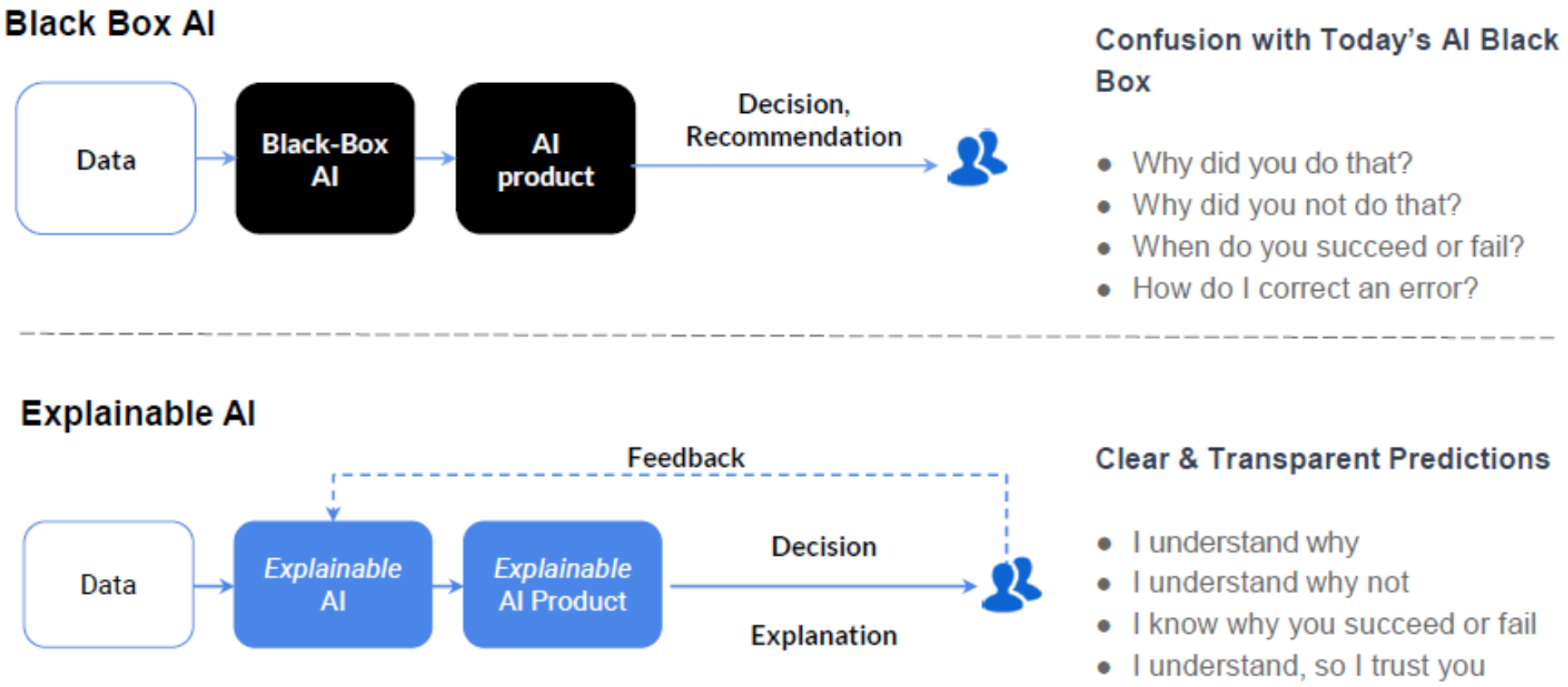
What is Interpretability?

- ***Interpretability*** is the ability to explain or to present in understandable terms to a human
- There is no clear answers in psychology to:
  - What constitutes an explanation?
  - What makes some explanations better than the others?
  - When are explanations sought?
- ***Explainable ML*** refers to methods and techniques in the application of ML systems such that the results of the solution can be understood by human experts and users
  - It opposes the concept of black-box models in ML, where it is not obvious why the model arrived at a specific decision

Slide credit: Hima Lakkaraju - Interpretability and Explainability

What is Interpretability?

- Black-box AI versus explainable AI

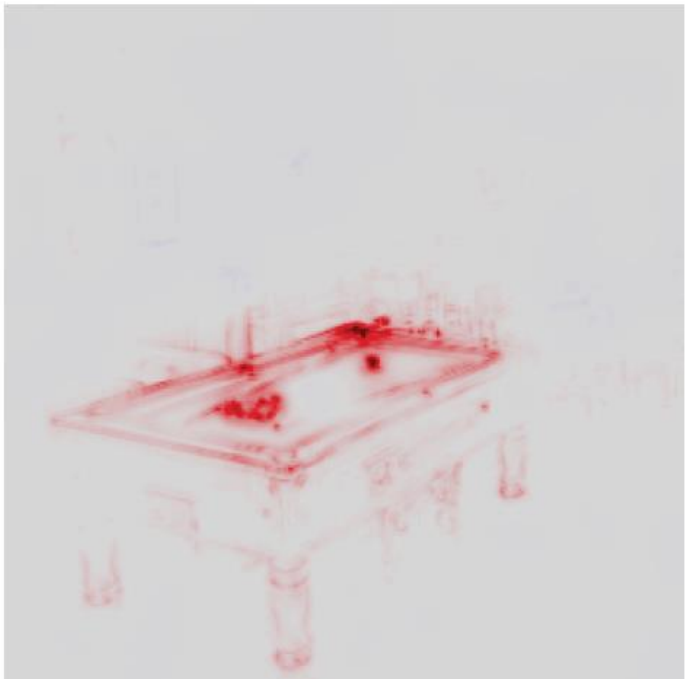


Picture from: KDD 2019 Tutorial, Explainable AI in Industry

Explainability in ML

- Explainability in ML can help to determine the most important features used by a model when making a prediction
  - E.g., the shown saliency map indicates the regions in the image that contributed the most to the classification by the model as "pool table"

True class label: pool table
Model prediction: pool table

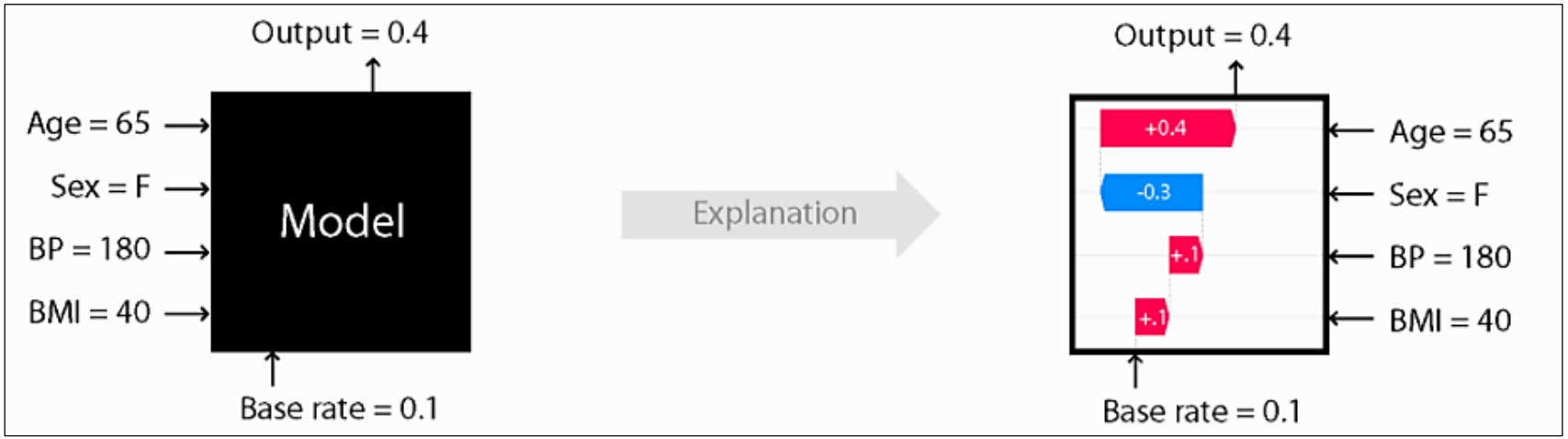


Saliency map, explains the model prediction

Picture from: Samek (2019) Meta-Explanations, Interpretable Clustering & Other Recent Developments.

Explainability in ML

- *Feature relevance* (or *feature attribution*) is also applied with non-image data for quantifying the influence of each input variable toward the model decision
  - E.g., the figure shows explainability via feature importance of an ML model for determining the interest rate of a bank loan



Picture from: Lundberg – SHAP documentation: https://shap.readthedocs.io/en/latest/

Concerns by Stakeholders

- Concerns regarding explainable AI faced by various ***stakeholders***
  - How does a model work?
  - What is driving decisions?
  - Can I trust the model

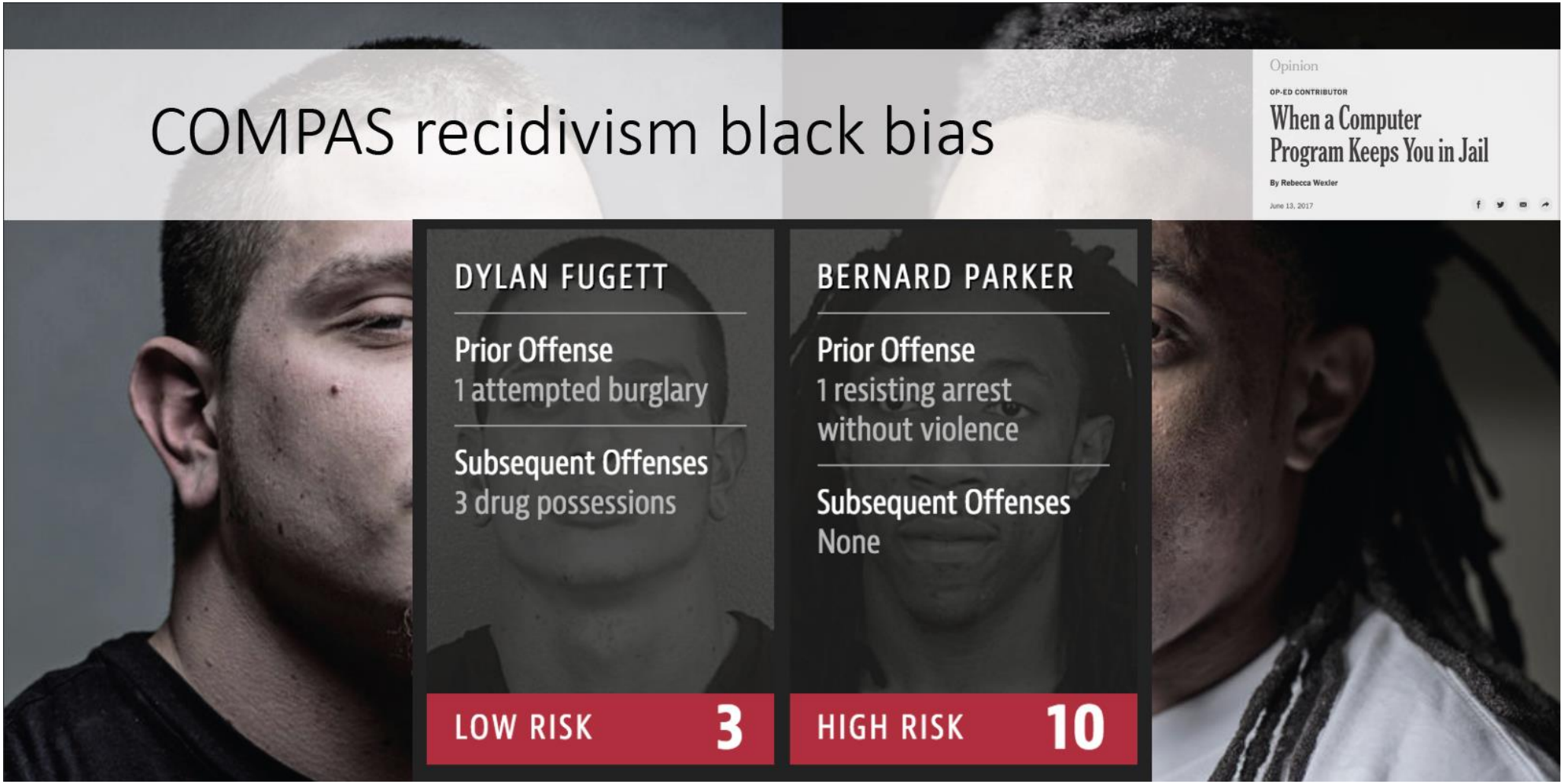
Picture from: Belle (2020) Principles and Practice of Explainable Machine Learning

Concerns when Deploying ML

- Correctness
  - Did only the variables of interest contribute to the decision, and not spurious patterns and correlations?
- Robustness
  - Is the model sensitive to minor data perturbations, or missing and/or noisy data?
- Bias
  - Are we aware of any data-specific biases that unfairly penalize groups of individuals?
- Improvement
  - Can the model be improved, e.g., via enhanced training data or feature space?
- Transferability
  - Can the model trained for one application domain be applied to another domain?
- Human comprehensibility
  - Can we explain the model's decision-making algorithm to a human domain-expert, or to a lay person?

ML Bias

- Example of a biased ML algorithm



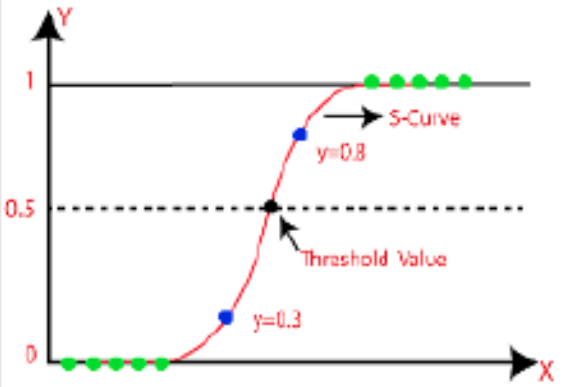
Picture from: Freddy Lecue, XAI – Explanation in AI

Transparency

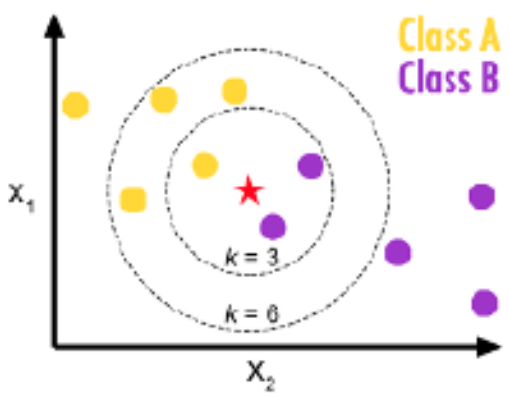
- *Transparency* stands for a human-level understanding of the inner workings of the model
- Transparent models are characterized by:
  - Simulatability – the model can be simulated by a human
  - Decomposability – the model can be broken down into parts (e.g., inputs, parameters, computations) that can be explained
  - Algorithmic transparency – the model allows to understand the procedure that it goes through in order to generate its output
- Broadly, we can think of ML models as:
  - *Transparent (interpretable) models*
    - Simpler models, easier to understand
    - E.g., decision trees, linear regression, logistic regression, rule-based models (if-else), *k*-nearest neighbors, Bayesian networks, generative additive models
  - *Opaque (black-box) models*
    - Nontransparent, difficult to understand
    - E.g., deep learning models, random forests, support vector machines, ensemble models

Transparent Models

Logistic regression



K Nearest Neighbours

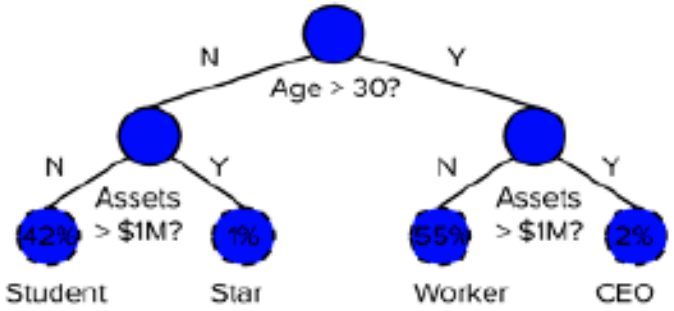


Rules

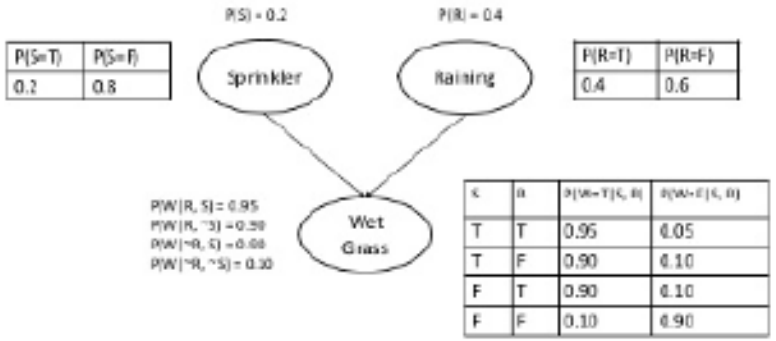
If — Income > £20k p.a. → Accept

Else — Income <= £20k p.a. → Reject

Decision Tree



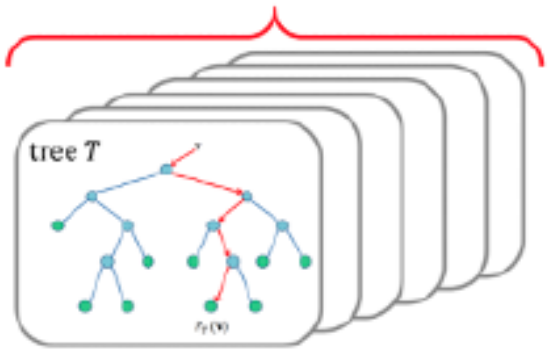
Bayesian Network



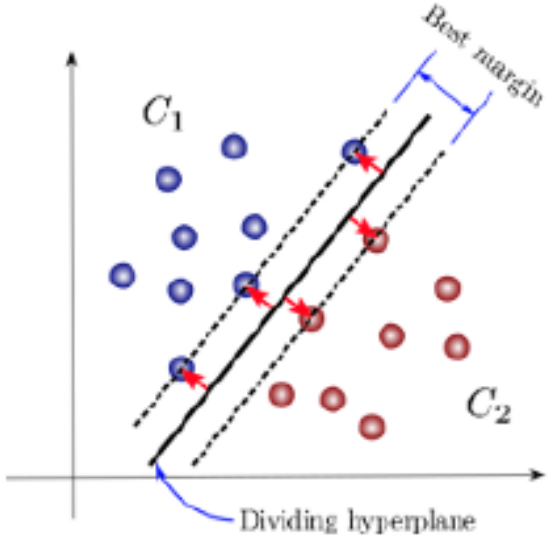
Picture from: Belle (2020) Principles and Practice of Explainable Machine Learning

Opaque Models

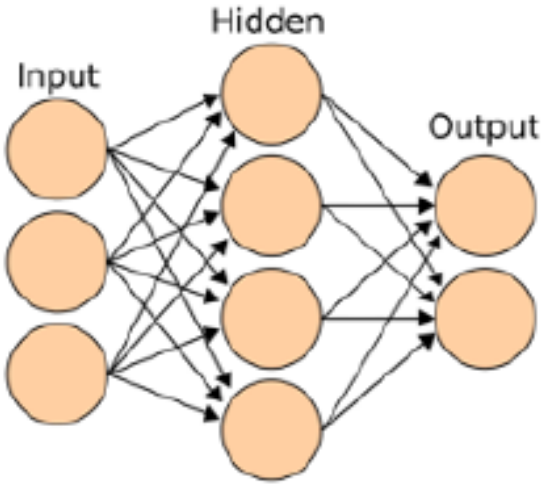
Random forest



Support Vector Machine



Neural network

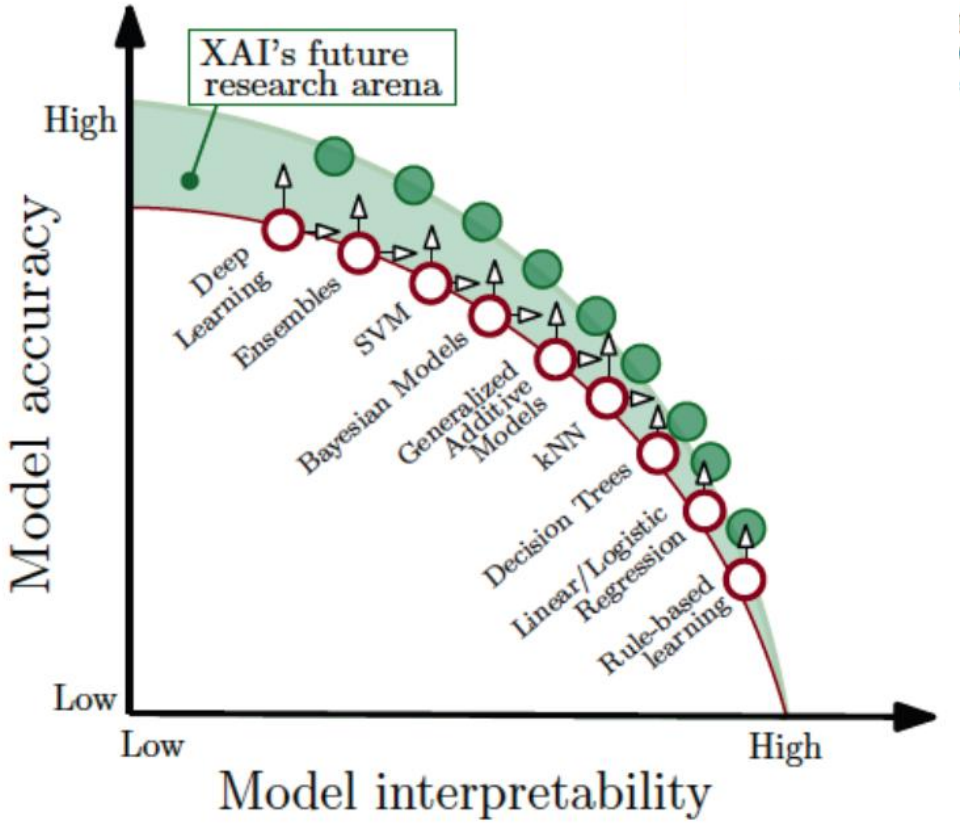


The challenge in each case is explaining how it really works and makes decisions!

Picture from: Belle (2020) Principles and Practice of Explainable Machine Learning

Interpretability vs Accuracy

- Currently, the best performing models in terms of accuracy are the least interpretable
  - Future research areas based on explainable modelling approaches offer a potential to enhance the interpretability of most ML models

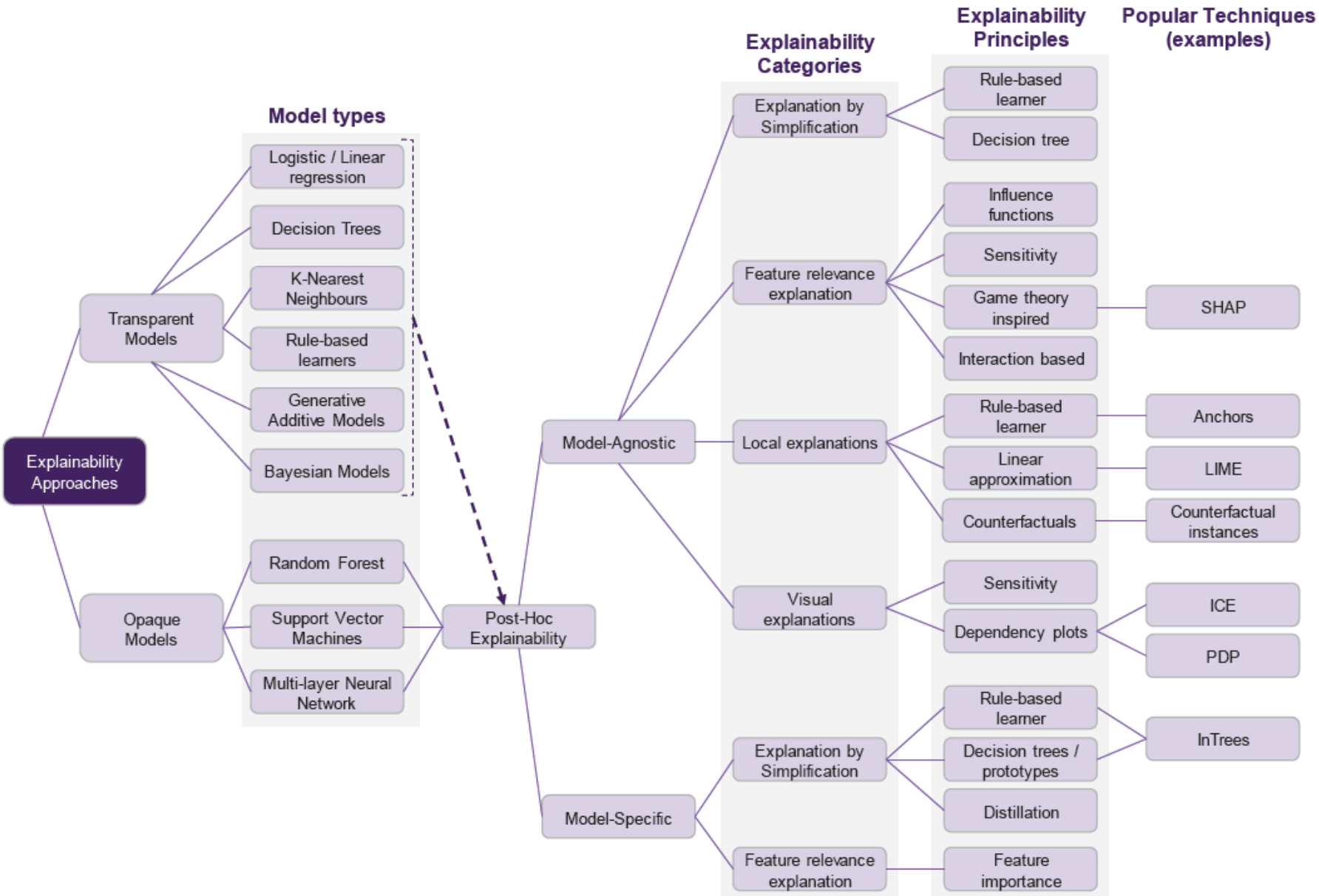


Picture from: Arietta (2019) – Explainable Artificial Intelligence (XAI)

Transparent vs Opaque Models

- At present , there are two major alternatives to achieving interpretability:
  1. ***Via transparent models,*** employ inherently interpretable models
     - They offer simplicity, since the explanation is embedded in the model
       - E.g., the weight coefficients **w** in the linear model $w_0 + w_1 f_1 + w_2 f_2 + \cdots + w_n f_n$ indicate which features **f** contributed the most to the decision
       - By following the branches in a decision tree model from the final outcome to the inputs, it is fairly easy to realize how the model made the predictions
       - Based on the applied if-then logic in a rule-based system, it is possible to understand the path from the inputs to the model prediction
     - The model simplicity can result in lower performance than more complex, but non-transparent models
       - In a critical application, a transparent model with lower accuracy may be acceptable
  2. ***Post-hoc explainability***: employ a post-hoc step to explain the predictions by opaque models
     - First train the opaque black-box model, and afterward apply another method to explain its decision-making process
     - Explanation by important features, concepts, influential examples
     - Downside: one extra step is needed to explain the black-box model

Map of Explainability Approaches



Picture from: Belle (2020) Principles and Practice of Explainable Machine Learning

Model-agnostic Explainability

- *Model-agnostic explainability*
  - Explainability techniques and methods that can be applied to explain any ML method
  - They don't depend on the internal architecture of the model that needs to be explained
  - Instead, they operate by relating the inputs of the model to its outputs
- *Model-specific explainability*
  - Explainability techniques and methods that are designed for explaining one or several types of ML models
  - These methods cannot be used for explaining any type of ML models

Type of Explanations

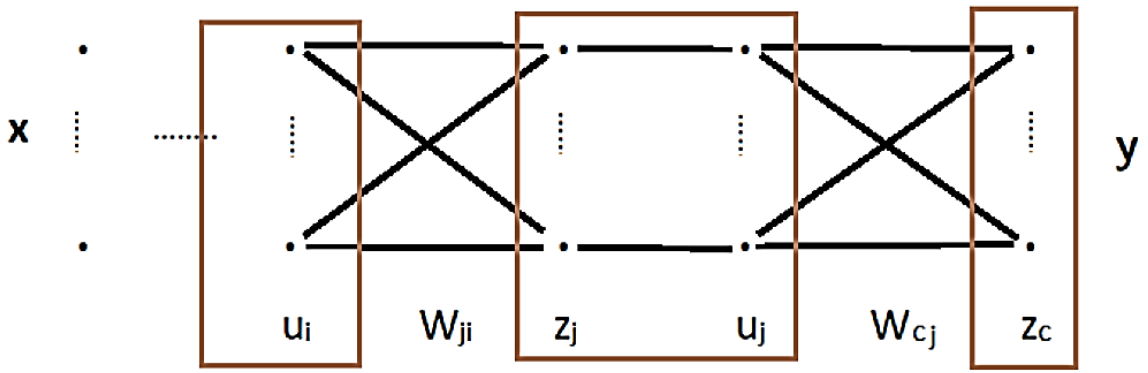
- **Pixel-level explanations**
  - Vanilla BackProp, Guided BackProp, Occlusion Maps, CAM, Grad-CAM, Guided Grad-CAM, Integrated Gradients, Layer-wise Relevancy Propagation
- Feature-level explanations
- Concept-level explanations
- Instance-level explanations

Vanilla Backpropagation

- *Vanilla Backpropogation*
  - Simonyan (2014) – Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
- The approach uses backpropagation to visualize the gradients with respect to each pixel of an image
  - A saliency map is obtained that highlights the pixels that have the largest impact on the class score
    - Image and the corresponding saliency map for the top-1 predicted class

Vanilla Backpropagation

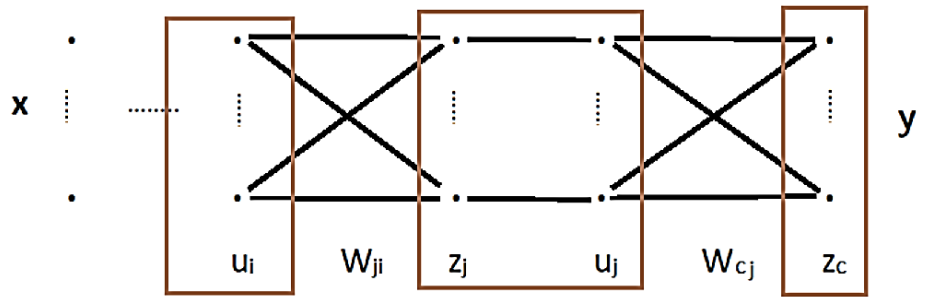
- The input to the DL model is an image $\mathbf{x} = (x_1, x_2, \ldots, x_d)$
- The neurons at layer j take the outputs from layer i (e.g., $z_j = \sum_i u_i W_{ji}$) and apply an activation function g to transform the inputs (e.g., $u_j = g(z_j)$)
- The vector of logits values for c-class classification is $\mathbf{z} = (z_1, z_2, \ldots, z_c)$
- The logits vector $\mathbf{z}$ is passed through softmax activations to produce the output probabilities $\mathbf{y}$



- Question: How important is the pixel x_i to the class score $z_c(\mathbf{x})$?
  - Sensitivity: How sensitive is the score $z_c(\mathbf{x})$ to changes in x_i?
  - Attribution: How much does x_i contribute to the score $z_c(\mathbf{x})$?

Vanilla Backpropagation

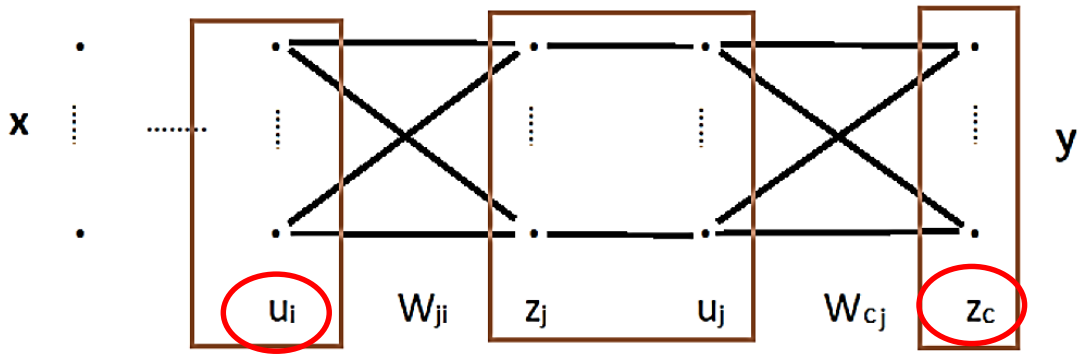
- Explaining the class score $z_c(\mathbf{x})$ of a trained DL model for an input image $\mathbf{x}$ in a post-hoc step
  - **Forward propagation**: Compute activations and the logits vector $\mathbf{z}$
  - **Backward propagation:** Compute the gradient of the class score z_c with respect to pixel x_i, $\frac{\partial z_c}{\partial x_i}$
    - For each neuron j on second-last layer: $\frac{\partial z_c}{\partial u_j} = W_{cj}$
    - For each neuron i on third-last layer: $\frac{\partial z_c}{\partial u_i} = \sum_j W_{ji} \frac{\partial u_j}{\partial z_j} \frac{\partial z_c}{\partial u_j}$
    - … the gradients are propagated to the input image to calculate $\frac{\partial z_c}{\partial x_i}$
- Recall that for model training, the gradient of the loss with respect to the weights is used for learning the optimal weight values, i.e., $\frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial W_{ji}}$



Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

Guided Backpropagation

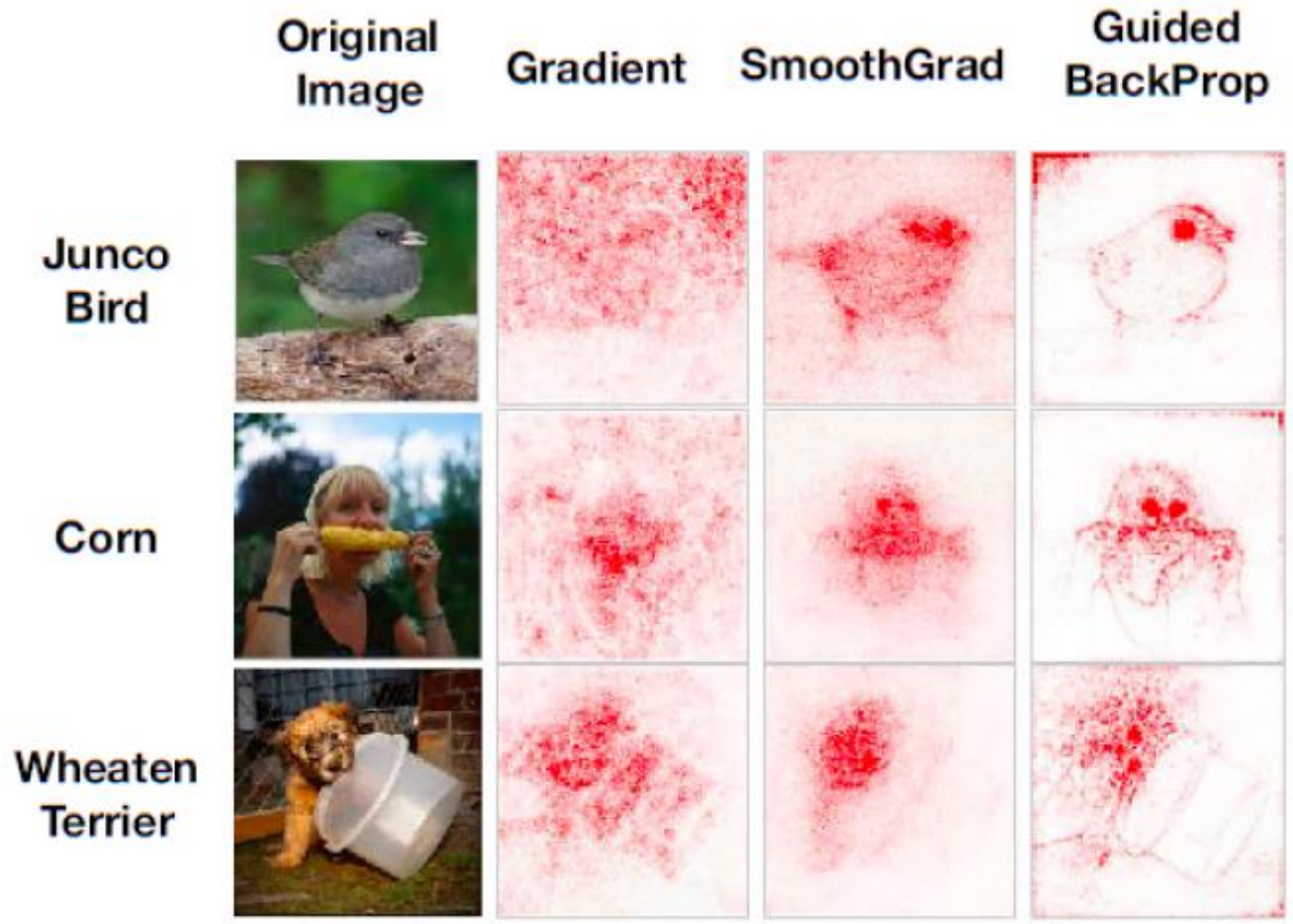
- *Guided Backpropagation*
  - Springerberg (2015) – Striving for Simplicity: Thee All Convolutional Net
- It expands the gradient backpropagation approach
  - If the gradient of a neuron is negative ($\frac{\partial z_c}{\partial u_j} < 0$), it contributes negatively to the gradients of the neurons in the lower layers
  - The guided backpropagation approach ignores negative gradients in the backpropagation step
    - A ReLU function is used for this purpose: $\frac{\partial z_c}{\partial u_i} = \sum_j W_{ji} \frac{\partial u_j}{\partial z_j} ReLU\left(\frac{\partial z_c}{\partial u_j}\right)$
    - Compare to Vanilla Backpropagation: $\frac{\partial z_c}{\partial u_i} = \sum_j W_{ji} \frac{\partial u_j}{\partial z_j} \frac{\partial z_c}{\partial u_j}$



Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

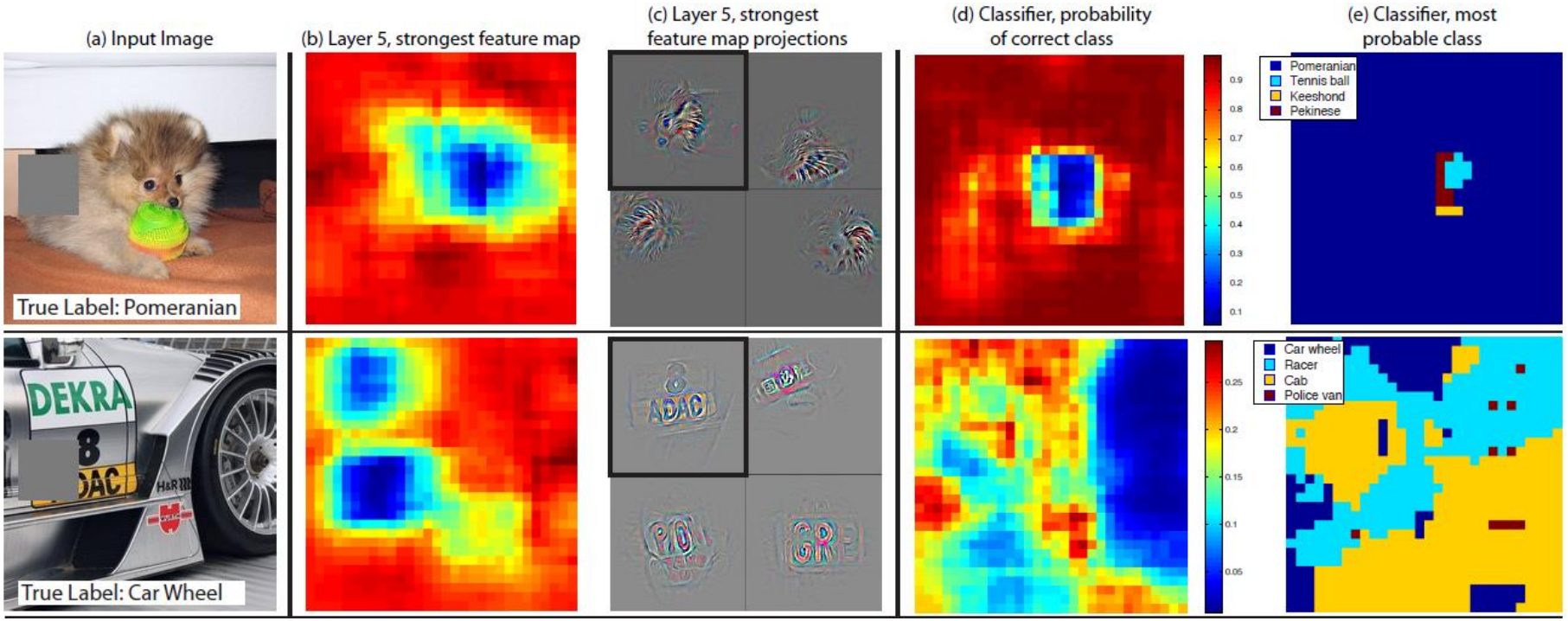
Guided Backpropagation

- Guided BackProp produces sharper saliency maps than Vanilla Gradient
  - Figure from Adebayo (2018) – Sanity Checks for Saliency Maps



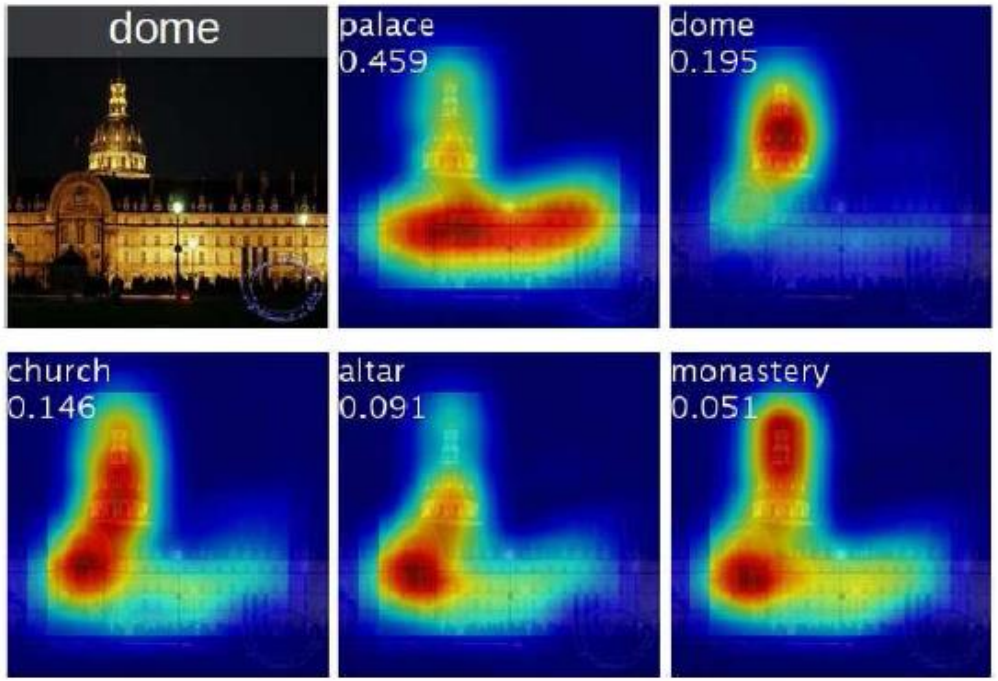
Occlusion Maps

- ***Occlusion maps***, also known as ***Perturbation Methods***
  - Zeiler (2014) – Visualizing and Understanding Convolutional Networks
- Systematically occlude different portions of the input image with a grey square, and monitor the output of the classifier
  - E.g., in the top image, the strongest feature is the dog's head
- Occlusion maps is an older and computationally expensive method



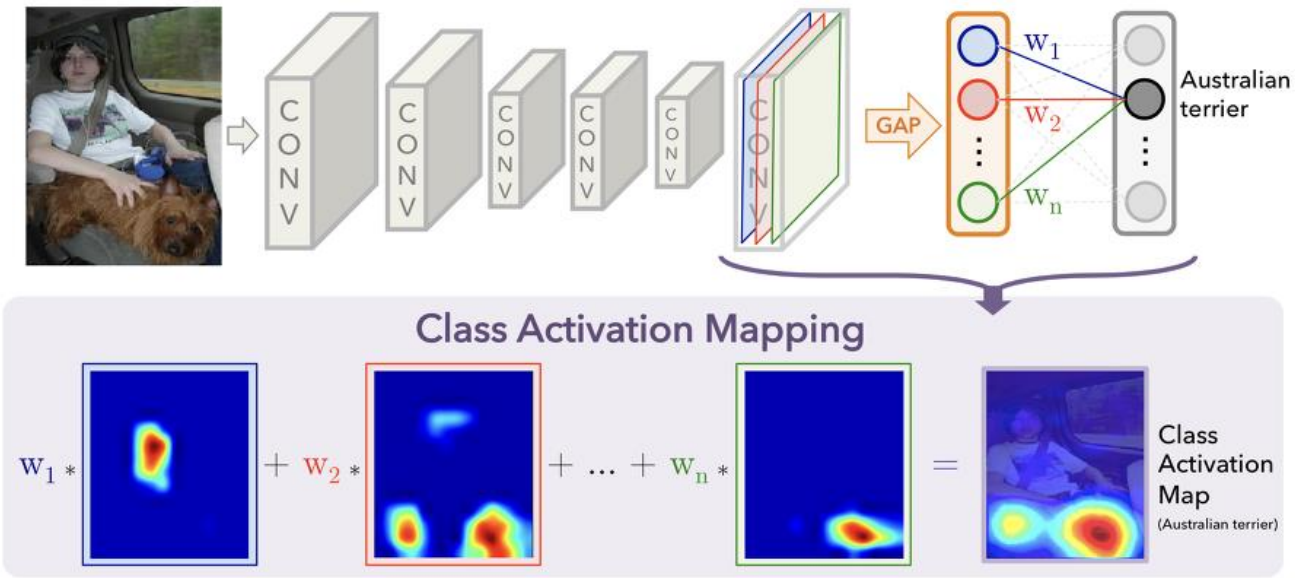
CAM

- *CAM (Class Activation Mapping)*
  - Zhou (2016) – Learning Deep Features for Discriminative Localization
- CAM produces a heatmap for the pixels that activate the most model's prediction of a specific class of objects
  - E.g., heatmaps for the top 5 predicted classes for the image with a true label 'dome'
    - Note that the 'palace' prediction focuses on the lower flat part of the building, while the 'dome' prediction focuses on the upper part



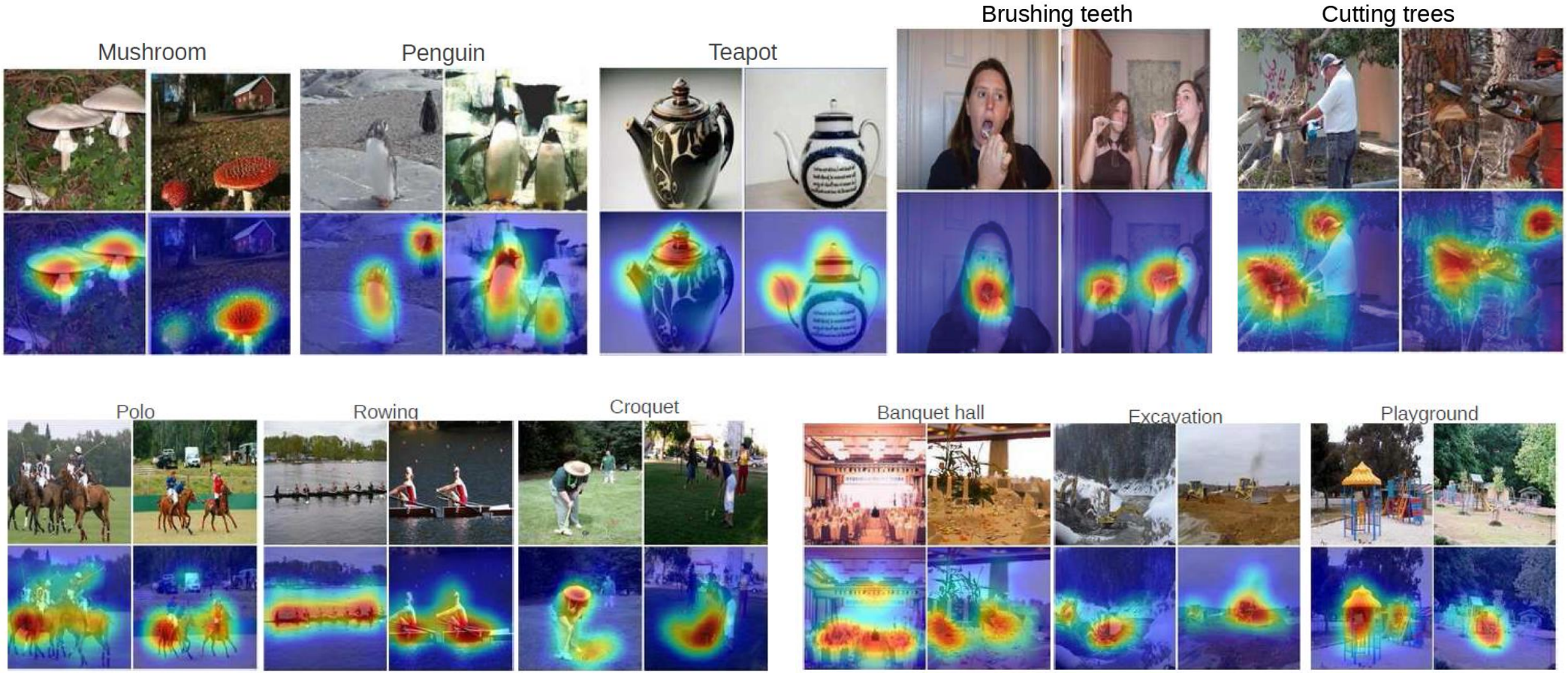
CAM

- Let $A^k = [a_{ij}^k]$ denotes the k-th feature map of the last convolutional layer in the network
- A global average pooling (GAP) layer is employed to calculate a value w_k for the feature map A^k, i.e., $w_k = \sum_{i,j} a_{ij}^k$
  - The outputs of the GAP layer $w_1, \ldots, w_k$ are fed to a softmax layer to output class predictions
- For a class c, the CAM heatmap is calculated as $\sum_k w_k^c A^k$



CAM

- Limitation: the last convolution layer has small resolution (typically between 7×7 and 28×28 pixels), therefore the produced heatmaps are coarse
- CAM applied to object recognition and action recognition tasks



Grad-CAM

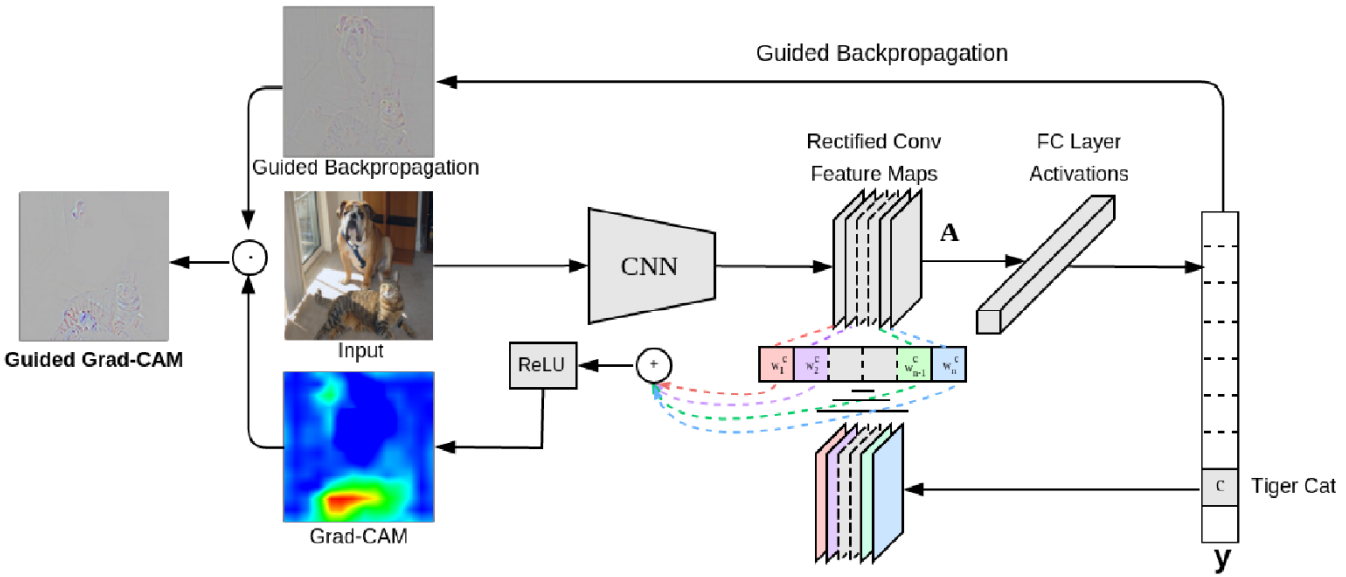
- *Grad-CAM*
  - Selvaraju (2017) – Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization
- Grad-CAM is an extension of the CAM approach
  - It employs gradient backpropagation to improve the heatmaps
    - Compare Grad-CAM to Guided BackProp for the classes 'Cat' and "Dog' in the image

(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog'

Grad-CAM

- Grad-CAM first computes the gradient of the logits for class c with respect to the feature maps $A^k = [a_{ij}^k]$ of the last convolutional layer in the network
  - I.e., the gradient is $\frac{\partial z_c}{\partial a_{ij}^k}$
- The importance of the feature map A^k is calculated as: $w_k = \sum_{i,j} \frac{\partial z_c}{\partial a_{ij}^k}$
- The Gad-CAM heatmap for a class c is calculated as: $ReLU\left(\sum_k w_k^c A^k\right)$
  - ReLU is used to select only the feature maps A^k that have positive influence on z_c

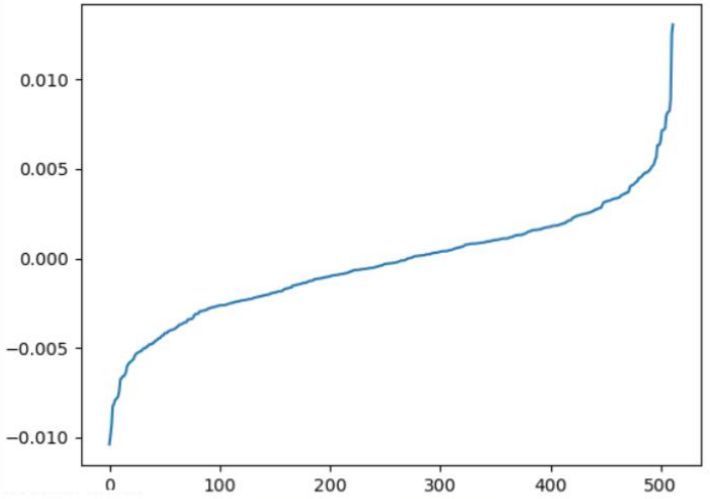


Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

Grad-CAM

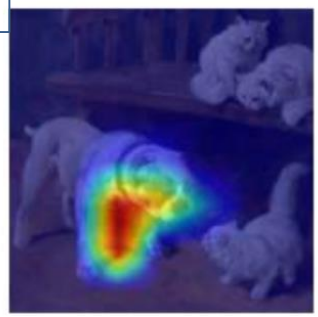
- Example, the values w_k for a network with 512 feature maps are shown below

Values w_k for $k = 0,1,..,511$



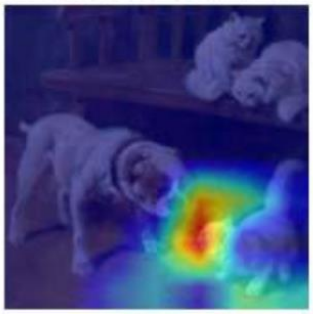
Final heatmap

Feature map A^0: 0
Value w_0: -0.01

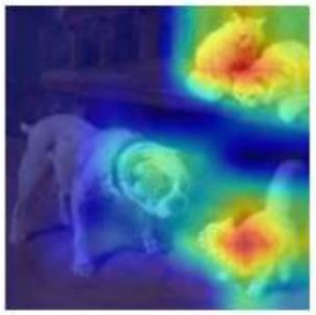


0: -0.010390018

255: -0.00027679346



306: 0.00041336069

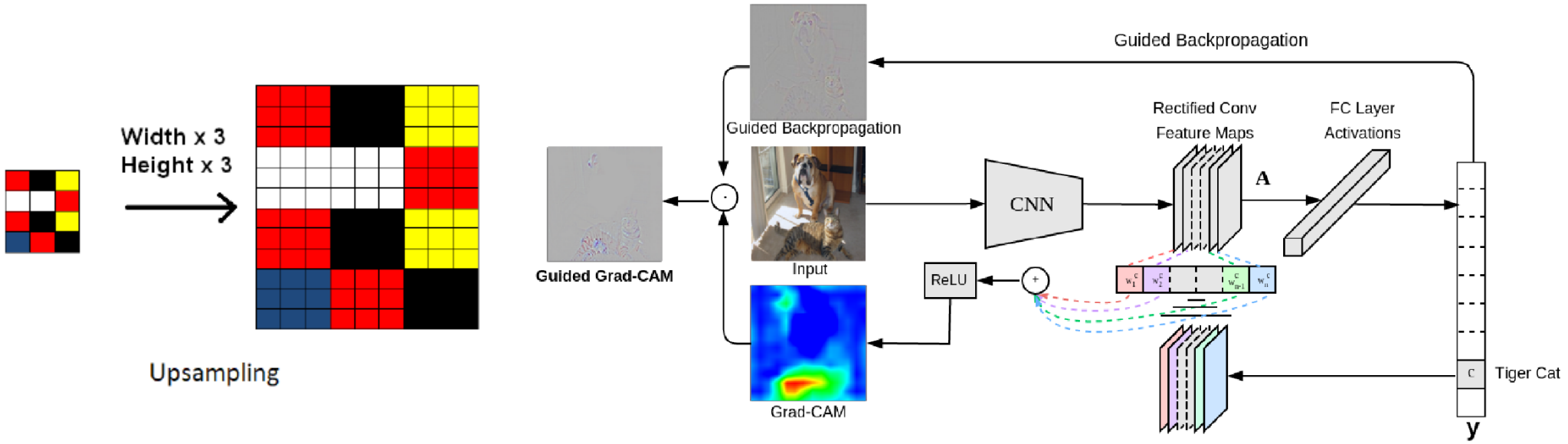


510: 0.012544246

511: 0.013066023

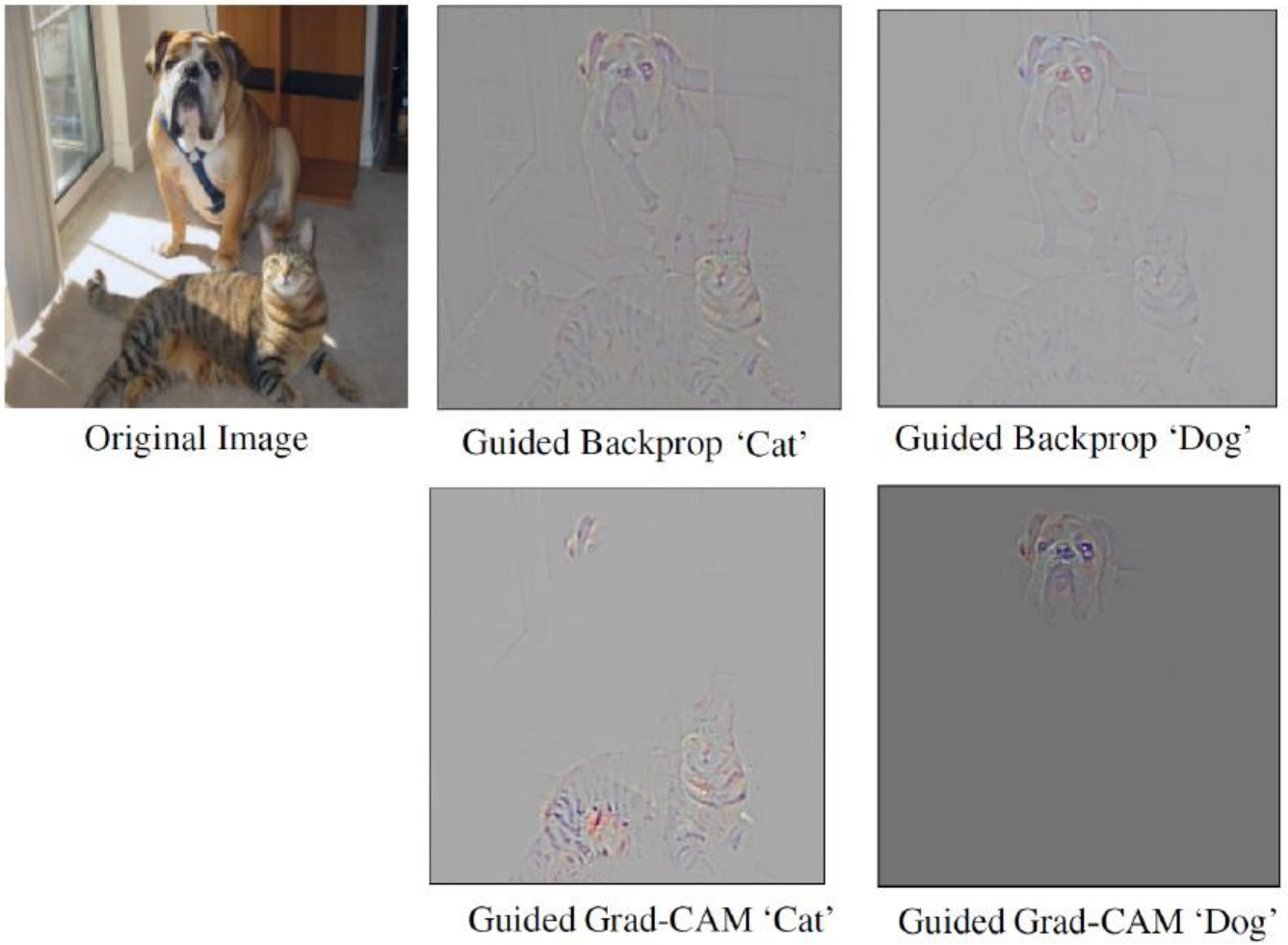
Guided Grad-CAM

- Like the original CAM method, the Grad-CAM heatmaps have smaller dimensions than the input image
- The authors of Grad-CAM proposed ***Guided Grad-CAM*** approach, in which the heatmaps produced by Grad-CAM are upsampled and multiplied with the saliency map from Guided BackProp

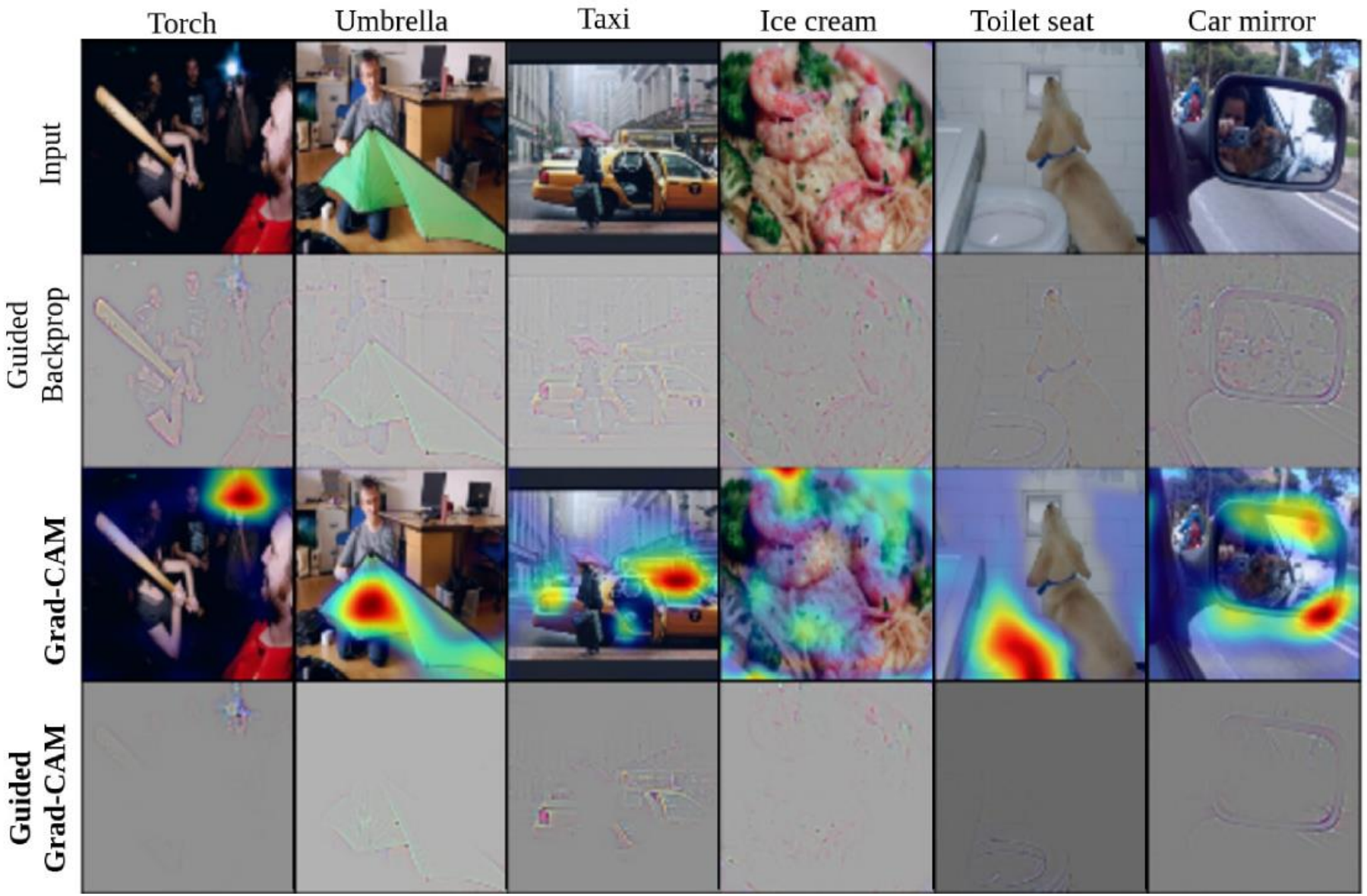


Guided Grad-CAM

- Guided Grad-CAM produces sharper and more discriminative saliency maps than Guided BackProp

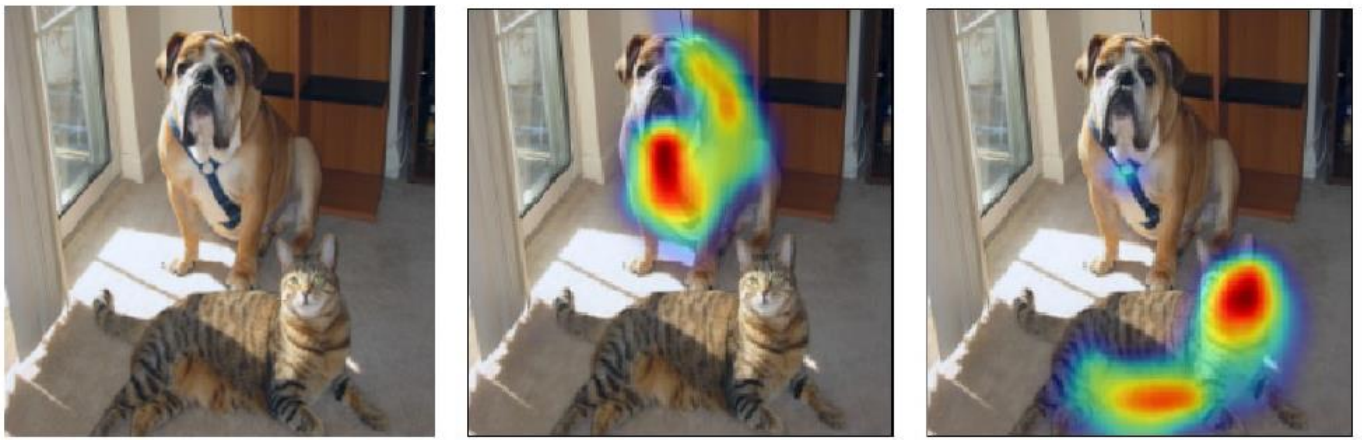


Guided Grad-CAM



Grad-CAM and Counterfactual Explanations

- To identify region that contributes positively to a classification
  - Importance weights: $w_k^c = \sum_{i,j} \frac{\partial z_c}{\partial a_{ij}^k}$ and heatmap: $ReLU\left(\sum_k w_k^c A^k\right)$
- To identify region that contributes negatively to a classification
  - Importance weights: $w_k^c = \sum_{i,j} -\frac{\partial z_c}{\partial a_{ij}^k}$ and heatmap: $ReLU\left(\sum_k w_k^c A^k\right)$
- Removing the negative region makes the classification more confident
  - The modified images are ***counterfactual explanations*** (emphasizing the regions that are opposite to the factual prediction)

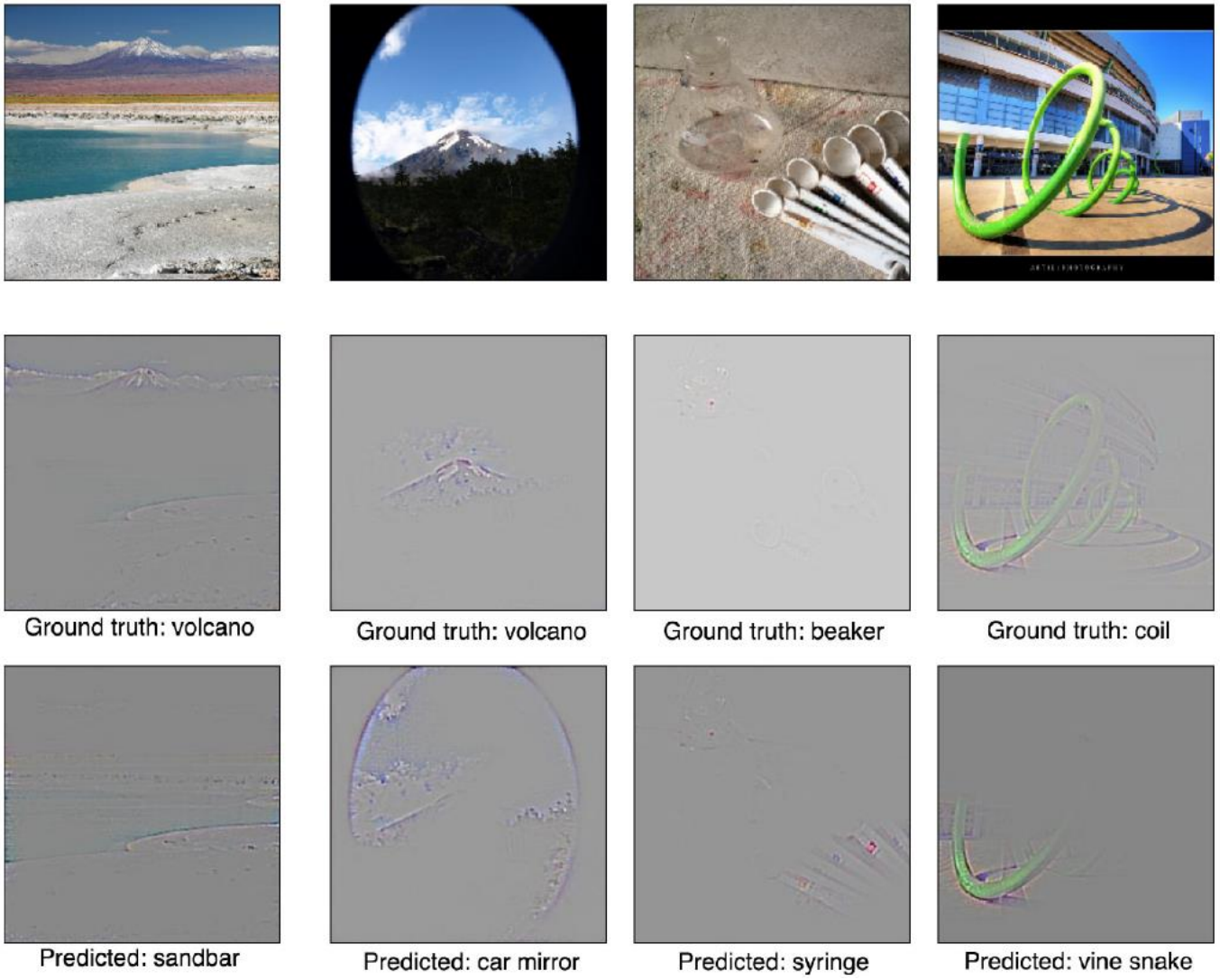


(a) Original Image (b) Cat Counterfactual exp (c) Dog Counterfactual exp

Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

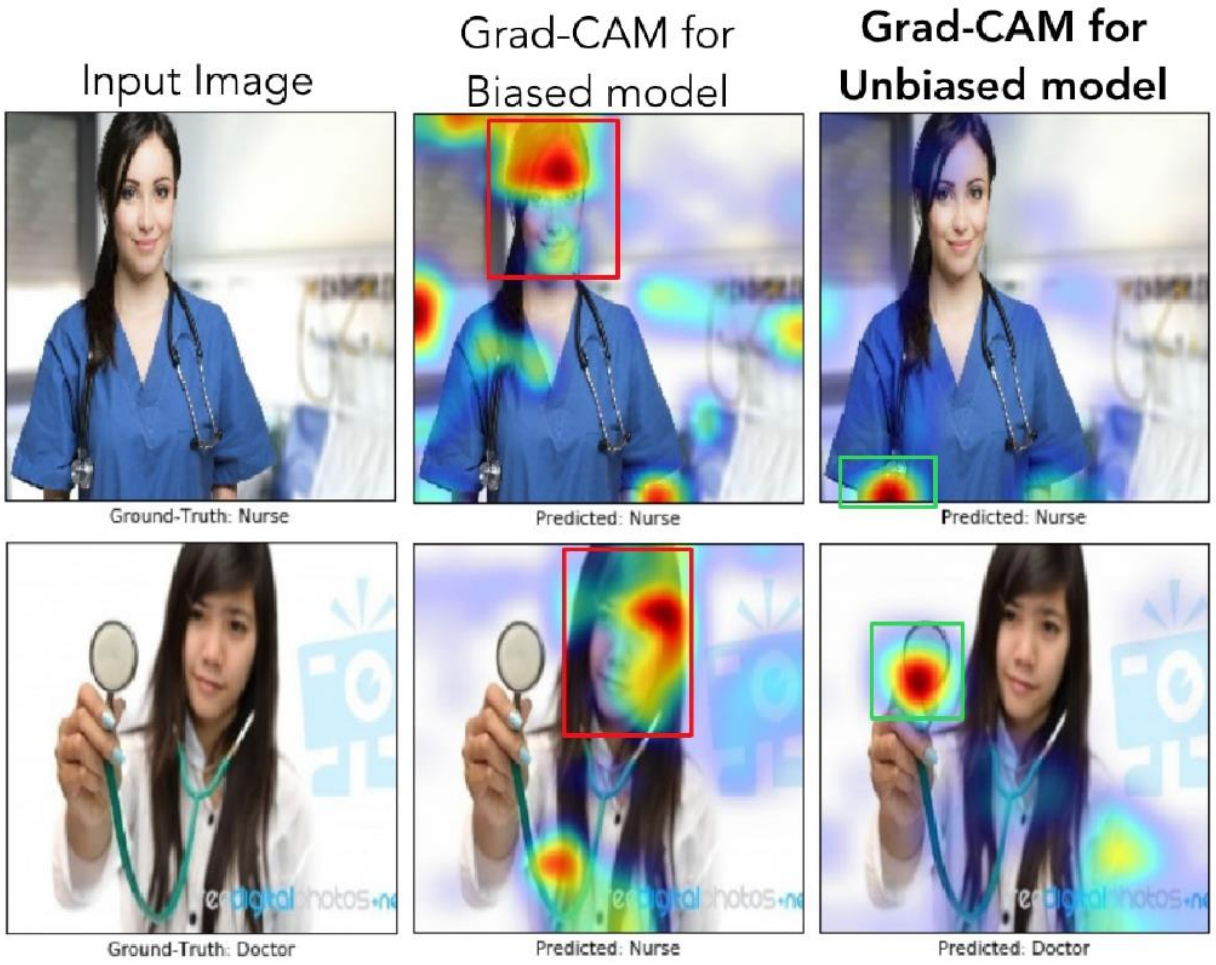
Guided Grad-CAM for Model Analysis

- Seemingly unreasonable predictions by models have reasonable explanations



Grad-CAM for Predicting Model Bias

- For a biased model, gender is strongly correlated with being a doctor or nurse
- For an unbiased model, gender is independent of being a doctor or nurse



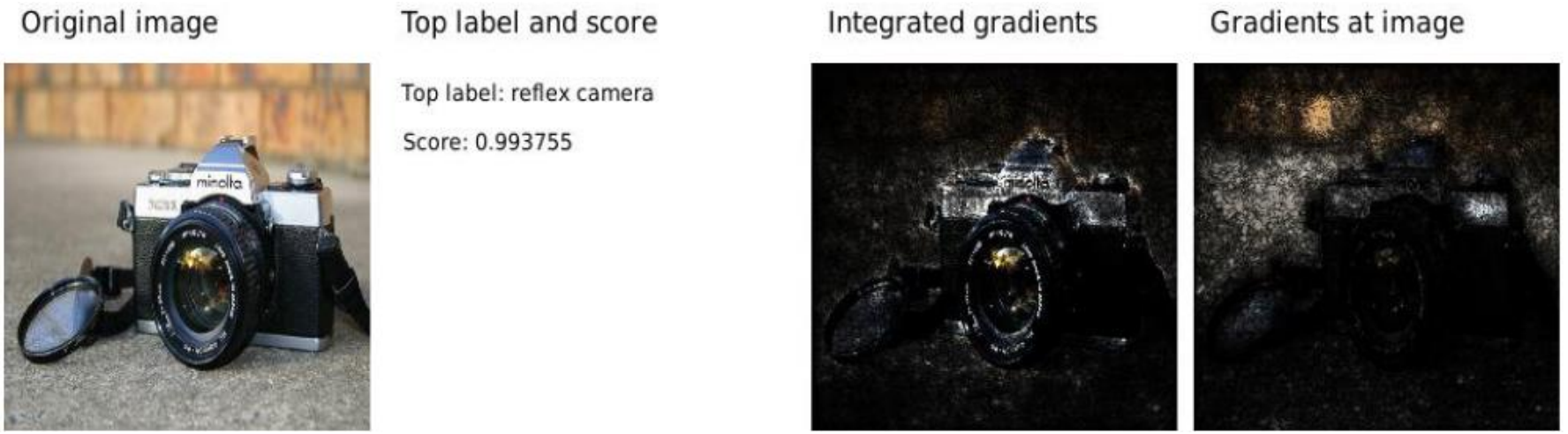
Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

Integrated Gradients

- ***Integrated Gradients***
  - Sundarajan (2017) - Axiomatic Attribution for Deep Networks
- Integrated gradients employs the integral of the gradients of a black-box model F along a straight-line path from a baseline $\mathbf{z}$ to an input instance $\mathbf{x}$
  - The integrated gradient for pixel i is calculated as

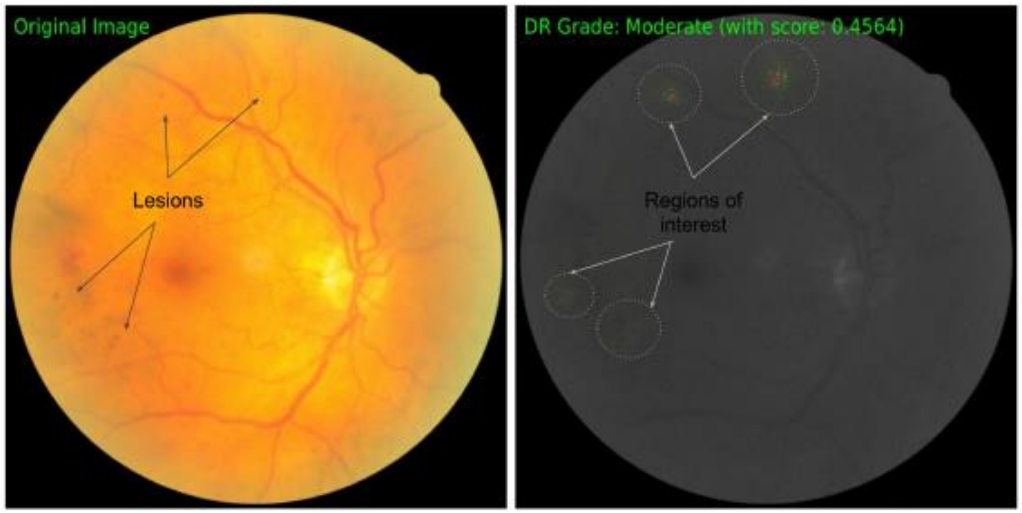
$$IG_i(\mathbf{x}, \mathbf{z}) = (\mathbf{x} - \mathbf{z}) \cdot \int_{\alpha=0}^{1} \frac{\partial F(\alpha \cdot \mathbf{x} + (1-\alpha) \cdot \mathbf{z})}{\partial x_i} d\alpha$$

  - E.g., in the figure below, the integrated gradients are sharper and more discriminative than the regular gradients



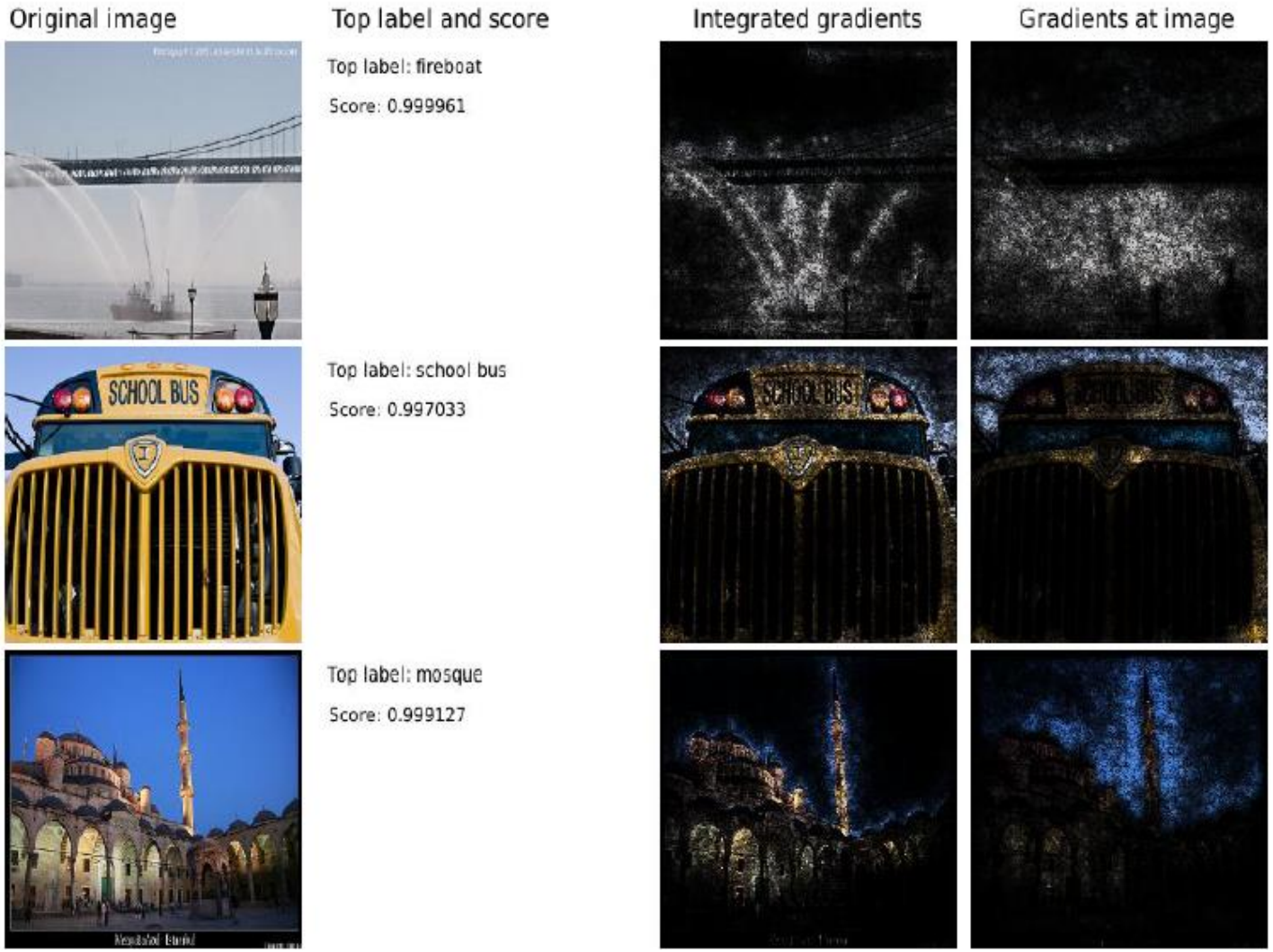
Integrated Gradients

- A key step in Integrating Gradients is selecting a baseline for the integration
  - Ideally, the baseline $\mathbf{z}$ is an information-less input for the model
  - This allows to better interpret the attributions as a function of the input
    - E.g., black image is used as a baseline for image models
    - E.g., empty text or zero-embedding vector for text models
- Integrated Gradients explain $F(\mathbf{x}) - F(\mathbf{z})$ in terms of input features
- E.g., application of IG for diabatic retinopathy prediction
  - The original image is on the left with the ground-truth lesions, and the on the right are the feature attributions pointing to the areas with lesions



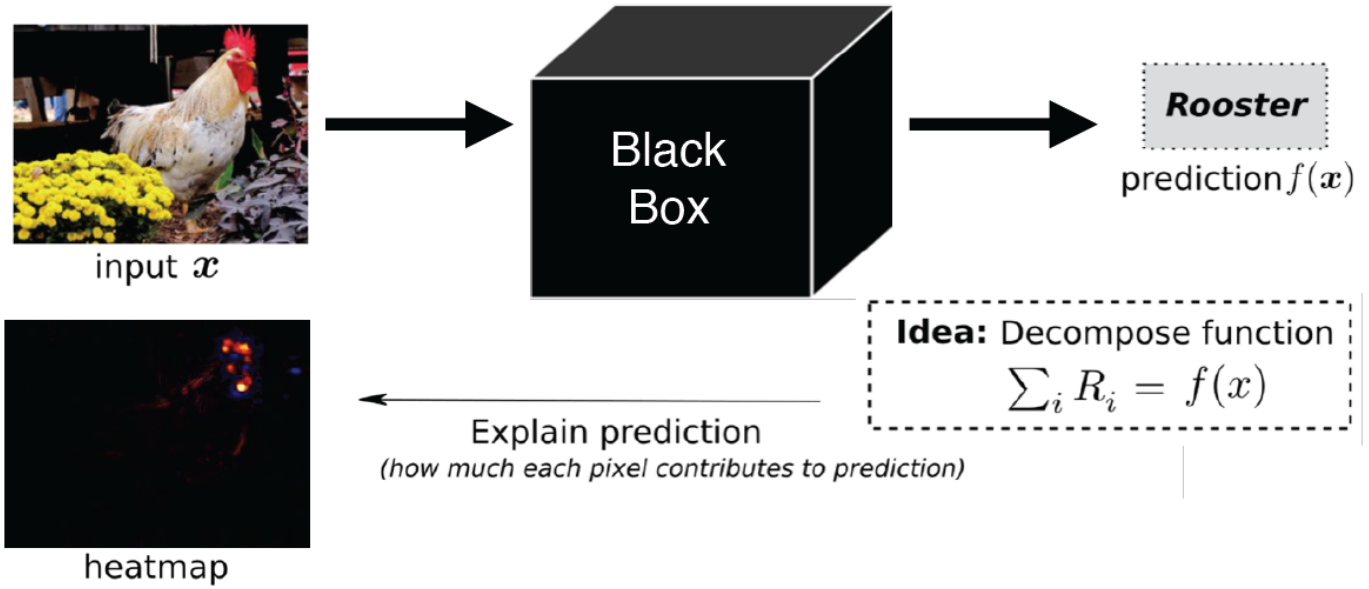
Integrated Gradients

- Examples



LRP

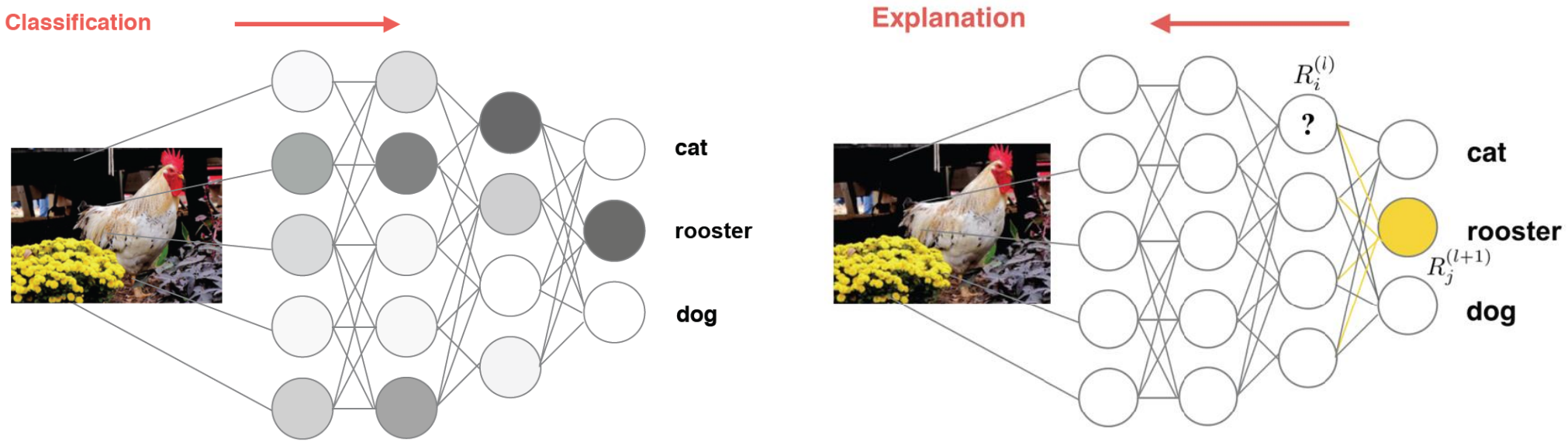
- *LRP (Layer-wise Relevancy Propagation)*
  - Bach (2015) - On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation
- LRP calculates the attribution (i.e., relevance, importance) of each pixel i in the input image $\boldsymbol{x}$ to the model prediction $f(\boldsymbol{x})$
  - The prediction $f(\boldsymbol{x})$ is propagated backward from the last layer toward the input layer
  - The sum of the relevance scores of all input pixels R_i is set equal to the predicted score for the class 'rooster'



Picture from: Samek (2019) Meta-Explanations, Interpretable Clustering & Other Recent Developments.

LRP

- Left image: the forward pass of the input image through the model results in the prediction 'rooster'
- Right image: the relevance score $R_j^{(l+1)}$ of the output neuron for the class 'rooster' is backpropagated from the top layer down to the input
  - The relevance score of neuron j in layer $l+1$, i.e., $R_j^{(l+1)}$ is decomposed into relevance scores of the neurons in the previous layer l, via $R_i^{(l)} = \sum_j \frac{x_i w_{ij}}{\sum_{i'} x_{i'} w_{i'j}} R_j^{(l+1)}$
    - w_{ij} are the weights between neurons i and j, and x_i is the input to neuron i



Picture from: Samek (2019) Meta-Explanations, Interpretable Clustering & Other Recent Developments.

LRP

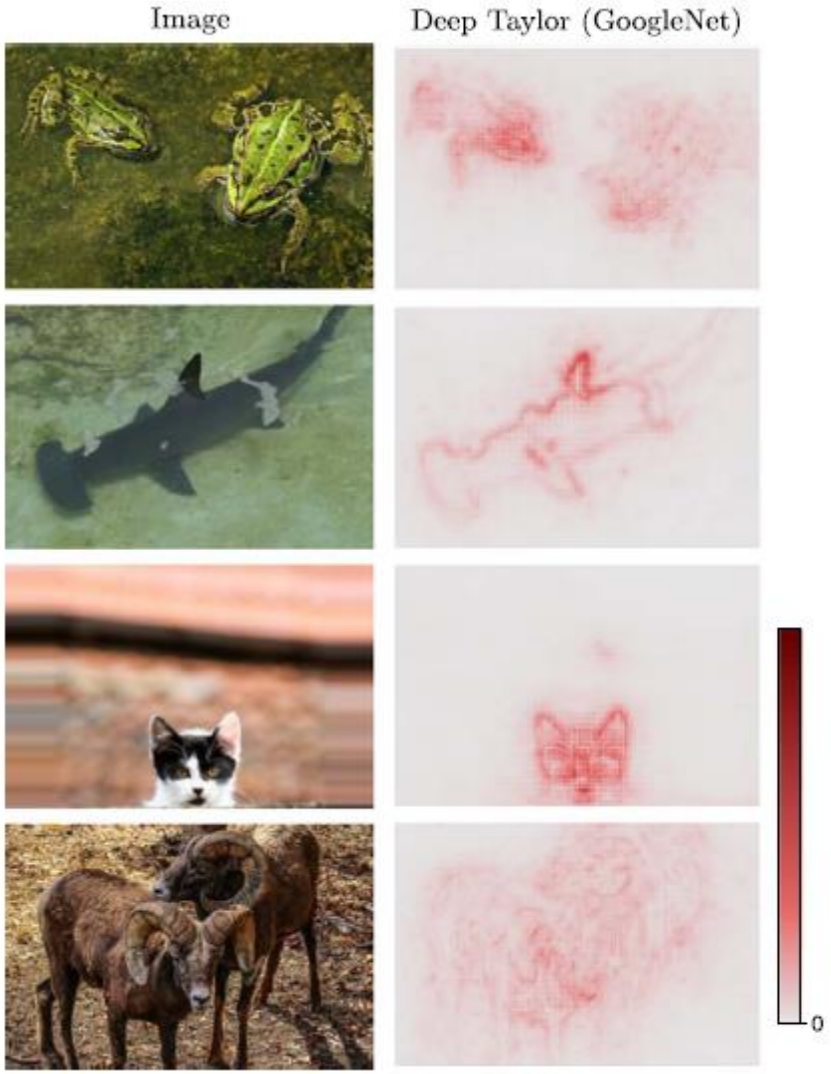
- The sums of the relevance scores over all layers is preserved

$$\sum_i R_i = \cdots = \sum_i R_i^{(l)} = \sum_i R_j^{(l+1)} = .. = f(x)$$

  - Meaning all layers (from the input layer of pixels R_i, all internal layers in the model $R_i^{(l)}$, to the neuron in the last prediction layer $f(x)$) have the same sum of the relevance scores
- The authors of LRP extended their approach by employing a modified Taylor expansion for the propagation of the relevance scores between the layers
  - Montavon (2017) - Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition
  - They refer to as ***deep Taylor decomposition***
  - The approach introduces rules for decomposing the relevance in different type of layers in deep NNs

LRP

- Images of 'frog', 'shark', 'cat', and 'sheep' and the corresponding LRP heatmaps



Application of Attributions

- Debugging model predictions
  - E.g., in the case of image misclassification, identify the pixels responsible for it
- Generating an explanation for the end-user
  - E.g., identify pixel attributions to a model's prediction to the end-user
- Analyzing model robustness
  - E.g., create adversarial examples using weaknesses surfaced by pixels' attributions
- Extract rules from the model
  - E.g., combine attributions to create rules for capturing the prediction logic of a drug screening model

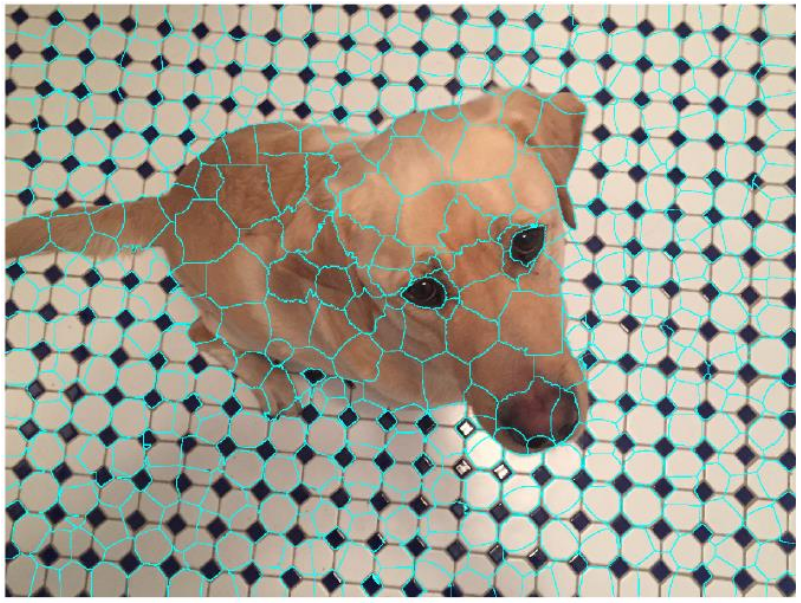
Slide credit: KDD 2019 Tutorial, Explainable AI in Industry

Type of Explanations

- Pixel-level explanations
- **Feature-level explanations**
  - LIME, Shapley values, SHAP
- Concept-level explanations
- Instance-level explanations

Feature-level Explanations

- ***Feature-level explanations*** refer to higher-level components in input data
  - For example, explaining regions in images at a higher abstraction than pixels
- Examples of ***features***:
  - Super-pixels in images
    - Collections of pixels, e.g., obtained by image segmentation as in the image below
  - Words in textual data
  - Input variables in tabular data



Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

LIME

- ***LIME (Local Interpretable Model-agnostic Explanations)***
  - Ribeiro (2016) – "Why Should I Trust You?": Explaining the Predictions of Any Classifier
- LIME is a model-agnostic explainability approach (it can work with any type of ML black-box models)
- The goal is to employ a simple ML model that can locally approximate the predictions by a black-box model
  - The simple ML model should identify interpretable representations of the data that are locally faithful to the black-box model
  - The interpretable representations should be understandable to a human user
  - Conversely, the input features (i.e., original representations of the data) may not be understandable to a human user

LIME

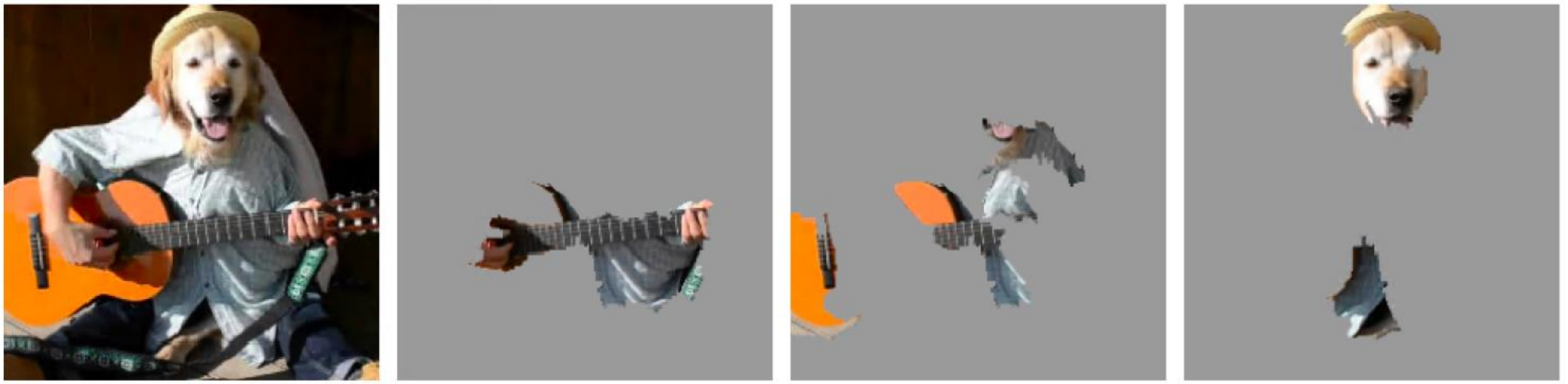
- Consider the image classification task
  - Original representation of an image **x** is the tensor of image pixels
  - A simplified representation of the image can consist of an array of super-pixels
  - Interpretable representation can indicate the presence or absence of super-pixels to describe a specific class label
    - Let assign a binary value $z \in \{0,1\}$ to each super-pixel, depending on whether the super-pixel is important for predicting the class c
    - The vector **z** of binary values over the set of all super-pixels is the interpretable representation
      - For this image, the vector **z** contain 1's for the important super-pixels for the class 'frog tree' and 0's for the non-important super-pixels for that class

Original representation
x

Interpretable representation
$\mathbf{z} = (0,0,1,0,1, \ldots, 0)$

LIME

- LIME explanation example
  - Explaining the top 3 class predictions by a trained Inception neural network: (1) electric guitar, (2) acoustic guitar, and (3) Labrador
  - The images (b), (c), and (d) show the super-pixels that contributed the most to each class prediction



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

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

LIME

- The classification black-box model that needs to be explained is denoted f
  - This can be any ML classifier model, such as neural network, random forest, ensemble models, etc.
  - The prediction by the black-box model for an input $\mathbf{x}$ is $f(\mathbf{x})$
  - E.g., the prediction is the probability that the input $\mathbf{x}$ belongs to a class c
- LIME uses a surrogate model g for explaining f
  - The surrogate model is an interpretable classifier model, such as a linear model, decision trees, rule lists model, etc.
  - The surrogate model g acts over the vector $\mathbf{z}$, related to the presence or absence of interpretable components in the input (using 0's and 1's for each component)
  - The prediction by the surrogate model is $g(\mathbf{z})$
- The prediction by the surrogate model $g(\mathbf{z})$ should be locally faithful to the prediction of the explained classifier $f(\mathbf{x})$ in the neighborhood of the input $\mathbf{x}$
  - I.e., the surrogate model should locally approximate the black-box model, and make approximately the same prediction: $g(\mathbf{z}) \approx f(\mathbf{x})$
  - The assumption is that the decision boundary of the black-box model is locally linear (although globally this boundary can be very complex)

LIME

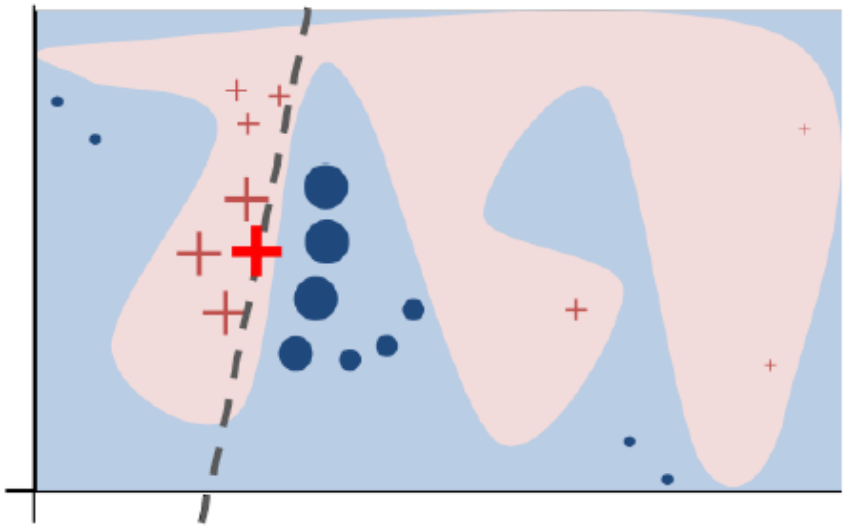
- The surrogate model g is chosen from a family of interpretable models G (linear or rule models, decision trees), and it is found by minimizing the cost function:

$$\arg\min_{g \in G} \mathcal{L}(f, g, \pi_{\mathbf{x}}) + \Omega(g)$$

- $\mathcal{L}(f, g, \pi_{\mathbf{x}})$ measures how faithful $g(\mathbf{z})$ is in approximating $f(\mathbf{x})$ locally, in the neighborhood of the input $\mathbf{x}$
- $\Omega(g)$ is a penalty term for the model complexity
  - I.e., a simpler model g from the family of models G is preferred, since simpler models are considered easier to interpret
- $\pi_{\mathbf{x}}$ is a distance function which measures the distance between the input $\mathbf{x}$ and instances $\mathbf{x}'$ in the neighborhood of $\mathbf{x}$
  - The authors proposed the following form: $\pi_{\mathbf{x}} = \exp(-D(\mathbf{x}', \mathbf{x})/\sigma)$
  - For images, ℓ_2 distance is used for the distance D in $\pi_{\mathbf{x}}$
- The cost function $\mathcal{L}(f, g, \pi_{\mathbf{x}}) + \Omega(g)$ is approximated by drawing multiple samples in the neighborhood of the input $\mathbf{x}$

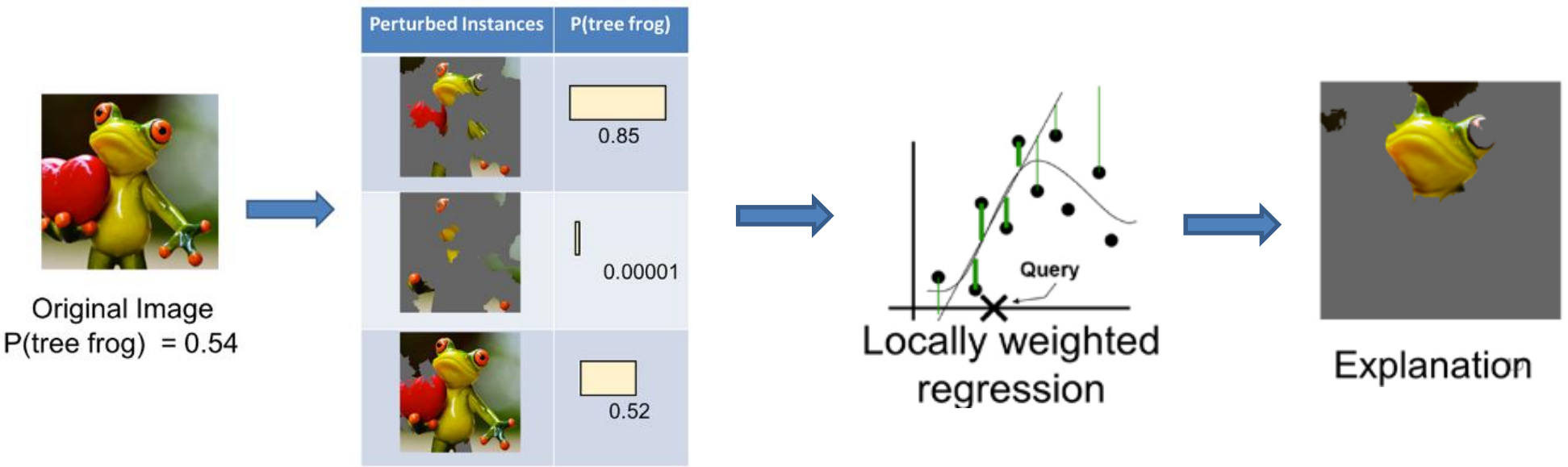
LIME

- The intuition behind LIME is shown in the figure
  - The black-box model f has a complex non-linear decision boundary
    - Depicted by the blue and pink regions: red crosses belong to one class of objects, and blue circles belong to another class of objects
  - The bold red cross is the instance **x** being explained
  - LIME samples instances in the neighborhood of **x**, it obtains class predictions using f, and weights the predictions based on the distance π_x to the input **x**
  - The dashed line is the learned linear explanation model g that is locally faithful to f
    - The global predictions by the simple linear ML model g are not assumed faithful to the predictions by non-linear model f



LIME

- LIME steps:
  1. Given an input image **x**, perturb it to create instances with slight modifications (e.g., by setting the color of some super-pixels to gray)
  2. Apply the black-box model f to calculate the probability of perturbed instances **x′**
  3. Select K super-pixels with top probabilities from the instances **x′** using LASSO
  4. Fit a simple (linear regression) model g to the perturbed instances **x′** (with the selected K super-pixels) weighted by their similarity distance π_x to the original input **x**
  5. Use the super-pixels with the highest coefficients to explain the model prediction

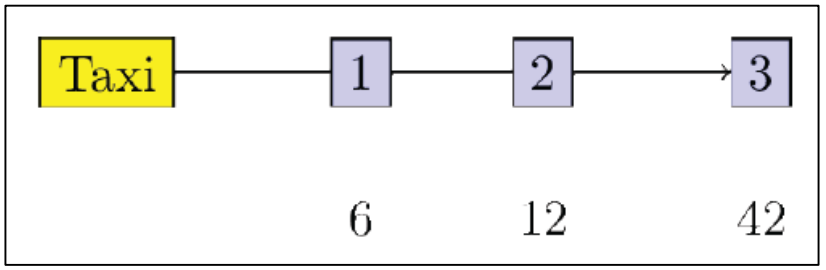


Shapley Values

- The ***Shapley values*** is a solution concept in game theory, named in honor of Lloyd Shapley
  - He introduced this approach in 1951, and was awarded the Nobel Prize in Economics for it in 2012
- Players in a game cooperate in a coalition and receive a certain profit from this cooperation
  - Players contribute differently to the total profit
  - For each cooperative game, the Shapley approach assigns to each player a portion of the total profit generated by the coalition
  - I.e., Shapley values tells us how to fairly distribute the "payout" among the players
- In ML, a prediction by a model can be explained by assuming that each feature of the data is a player in a game where the prediction is the payout
  - Shapley values calculate the importance (the contribution) of different data features in making a specific prediction by the model

Shapley Values

- **Example:**
  - Three persons share a taxi
  - The costs for the individual journeys are:
    - Person 1 journey: \$6
    - Person 2 journey: \$12
    - Person 3 journey: \$42
  - How much should each person contribute to the total cost of \$42?

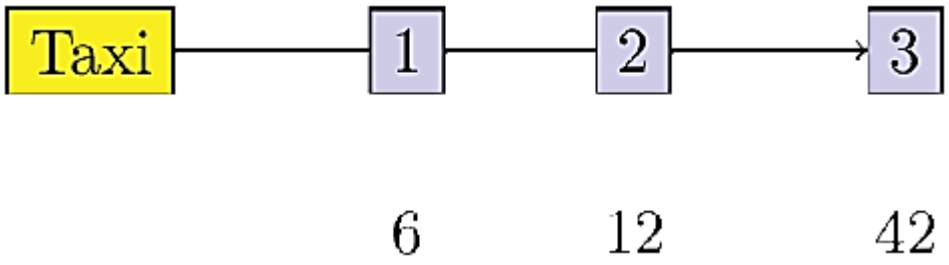


- Consider the problem as a cooperative game
  - Create the sets of all possible coalitions between the three persons (i.e., players)
  - For 3 players, there are $2^3 = 8$ possible sets of coalitions (i.e., subsets of players) S: $\{\emptyset\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$
  - For any coalition S, a characteristic function $f(S)$ assigns a payout value
    - The assigned values are shown on the next page

Shapley Values

- Table with the possible sets of coalitions S and the corresponding payout values of the characteristic functions $f(S)$ for each set

S	$f(S)$
{1}	6
{2}	12
{3}	42
{1, 2}	12
{1, 3}	42
{2, 3}	42
{1, 2, 3}	42



- How to divide the total cost of $42 among the three persons?
  - The next page describes the approach using Shapley values

Shapley Values

- The Shapley value for player i is calculated as:

$$\phi_i(f) = \frac{1}{M!}\sum_{\pi\in\Pi}\Delta_\pi^f(i) = \frac{1}{M!}\sum_{\pi\in\Pi}\left[f\left(S_\pi^i \cup \{i\}\right) - f\left(S_\pi^i\right)\right]$$

- M is the number of players
  - For the taxi example with 3 players, the factorial is $M! = 3! = 3 \cdot 2 \cdot 1 = 6$
- π is a permutation of the set $\{1, 2, .., M\}$
  - E.g., for the set of three players there are $3! = 6$ possible permutations
  - Π denotes the permutations: $\{1, 2, 3\}$, $\{1, 3, 2\}$, $\{2, 1, 3\}$, $\{2, 3, 1\}$, $\{3, 1, 2\}$, and $\{3, 2, 1\}$
- $\Delta_\pi^f(i)$ is the marginal contribution of player i to the Shapley value
  - It is calculated as the difference between the values of the characteristic function $f(S)$ when the player i is included $\left(f\left(S_\pi^i \cup \{i\}\right)\right)$ and when player i is excluded $\left(f\left(S_\pi^i\right)\right)$ from the coalition
- S_π^i is the set of predecessors of player i in π

Shapley Values

- Let's consider the following permutation of the set: $\pi = \{2, 3, 1\}$
  - For player 1: $i = 1$
    - The set of predecessors of 1 in $\{2, 3, 1\}$ is $S_\pi^1 = \{2, 3\}$
    - Similarly, $S_\pi^1 \cup \{i\} = S_\pi^1 \cup \{1\} = \{1, 2, 3\}$
    - The marginal contribution of player 1 is $\Delta_\pi^f(1) = f(S_\pi^1 \cup \{1\}) - f(S_\pi^1) = f(\{1, 2, 3\}) - f(\{2, 3\}) = 42 - 42 = 0$
      - The values for $f(\{1, 2, 3\})$ and $f(\{2, 3\})$ are read from the table on the previous page
  - For player 2: $i = 2$
    - The set of predecessors is $S_\pi^2 = \{\emptyset\}$, and $S_\pi^2 \cup \{2\} = \{2\}$
    - The marginal contribution of player 2 is $\Delta_\pi^f(2) = f(\{2\}) - f(\{\emptyset\}) = 12 - 0 = 12$
  - For player 3: $i = 3$
    - $S_\pi^3 = \{2\}$, and $S_\pi^3 \cup \{3\} = \{2, 3\}$
    - The marginal contribution is $\Delta_\pi^f(3) = f(\{2, 3\}) - f(\{2\}) = 42 - 12 = 30$

Shapley Values

- The table shows the marginal contributions $\Delta_\pi^f(i)$ by players 1, 2, and 3 for all 6 permutations of the set $\{1, 2, 3\}$
- The calculated Shapley values $\phi_i(f)$ for each player are shown at the bottom
  - Therefore, the fair way of sharing the taxi fare is for player 1 to pay \$2, player 2 to pay \$5, and player 3 to pay \$35

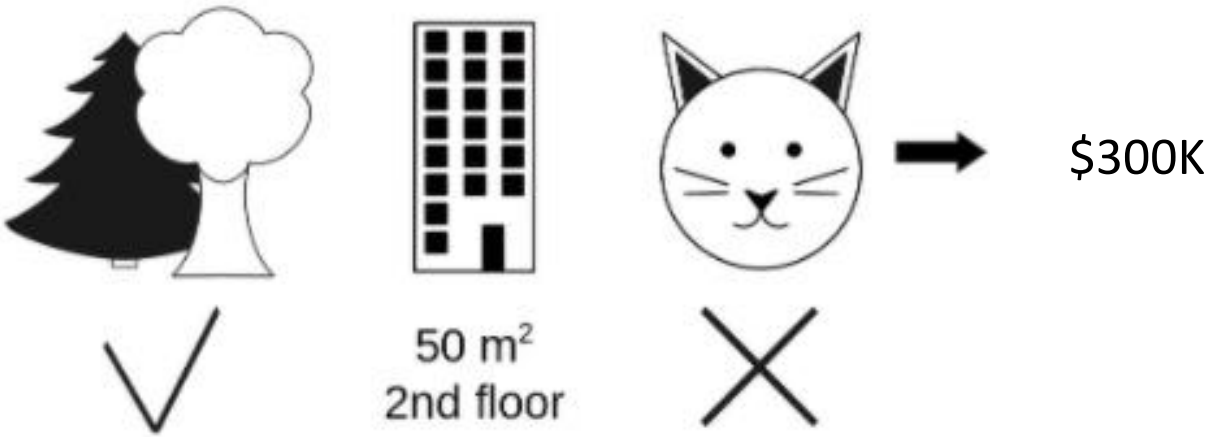
π	$\Delta_\pi^f(1)$	$\Delta_\pi^f(2)$	$\Delta_\pi^f(3)$
$\{1,2,3\}$	6	6	30
$\{1,3,2\}$	6	0	36
$\{2,1,3\}$	0	12	30
$\{2,3,1\}$	0	12	30
$\{3,1,2\}$	0	0	42
$\{3,2,1\}$	0	0	42
$\sum \Delta_\pi^f(i)$	*12*	*30*	*210*
$\phi_i(f) = \frac{1}{6}\sum \Delta_\pi^f(i)$	*2*	*5*	*35*

Shapley Values

- Shapley values are often used to explain the prediction $f(\mathbf{x})$ of a black-box ML model f on an input instance $\mathbf{x}$
- Each feature of $\mathbf{x}$ is regarded as a player, and the joint payout of all players (features) is $f(\mathbf{x})$
- We need to define a characteristic function $f_{\mathbf{x}}(S)$, where S is a subset of the features $\{1, 2, \ldots, M\}$, and M is the number of features
  - There are several approaches for calculating $f_{\mathbf{x}}(S)$, including:
    1. Sample independently a set of input examples $\mathbf{x}^1, \ldots \mathbf{x}^N$ and estimate $f_{\mathbf{x}}(S)$ as an average value of $f(\mathbf{x}^i)$ for the subset of features in S based on $f_{\mathbf{x}}(S) \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}^i)$
    2. Sample independently a set of input examples $\mathbf{x}^1, \ldots \mathbf{x}^N$ and estimate $f_{\mathbf{x}}(S)$ as an average value based on features not in the subset S, i.e., $f_{\mathbf{x}}(S) \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}^i_{\bar{S}})$, where $\mathbf{x}^i_{\bar{S}}$ denotes the subset of features in $\mathbf{x}^i$ that are not in the subset S
    3. Sample independently a set of input examples $\mathbf{x}^1, \ldots \mathbf{x}^N$ and set the features not in the subset S to a reference value, and estimate $f_{\mathbf{x}}(S)$ as $f_{\mathbf{x}}(S) \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}^{i,r}_{\bar{S}})$, where $\mathbf{x}^{i,r}_{\bar{S}}$ denotes the input $\mathbf{x}^i$ with the reference values for the subset of feature that are not in the subset S
       - For images, often 0 (black) intensity values of the pixels are used as reference value
       - For tabular data, reference value is either the data mean, median, or a representative data example

Shapley Values

- **Example:**
  - ML is trained to predict apartment prices
  - Four features are used as inputs for model training and prediction: (1) size, (2) floor, (3) park-nearby, and (4) cat-banned
  - For an apartment with a size of 50 m^2, located on the 2nd floor, has a park nearby, and cats are banned, the predicted value by the model is \$300K
  - The average prediction for all apartments in the dataset is \$310K
  - Goal: explain how much each of the four features of the apartment contributed to the difference in the price between the average \$310K and the predicted \$300K



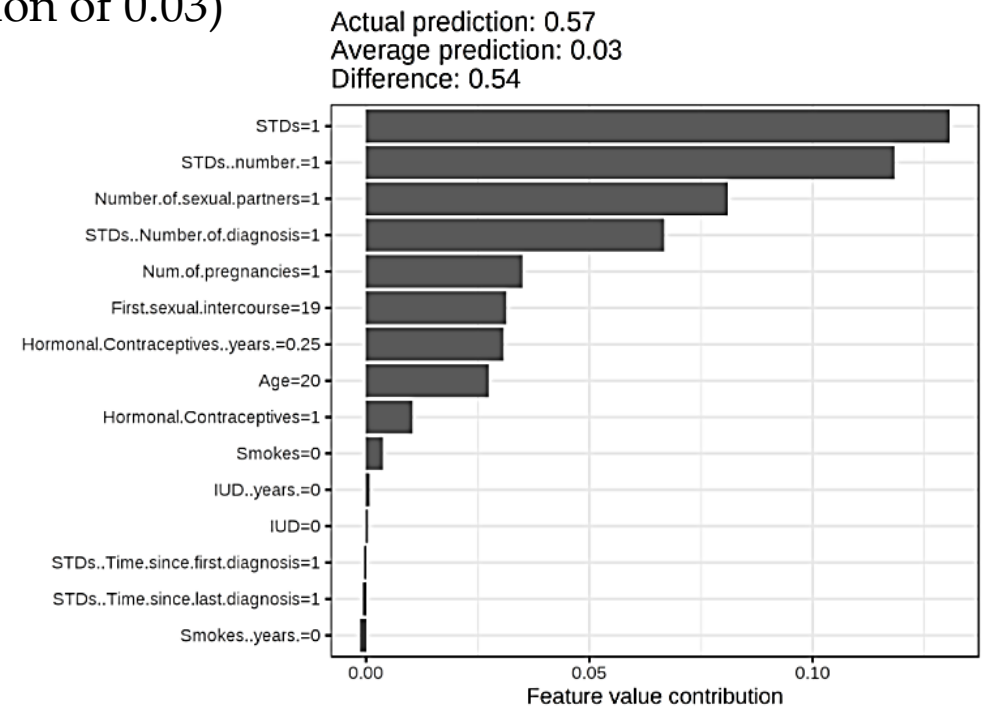
Picture from: https://christophm.github.io/interpretable-ml-book/shapley.html

Shapley Values

- To calculate the Shapley values, let's consider the contribution of the 'cat-banned' feature to the coalition (subset of features) of 'park-nearby' and 'size-50'
  - We randomly sample from the other apartments in the dataset and select a value for the 'floor-size' feature: for instance, the randomly drawn value is '1st-floor'
    - We predict the price of the apartment: for this combination of features, it is obtained \$310K
  - Next, we replace the feature with one randomly selected value from the set {'cat-banned', 'cat-allowed'}: for instance, the randomly drawn value is 'cat-allowed'
    - We predict the price for this combination of features, obtaining \$320K
  - The contribution of the 'cat-banned' feature is the difference $\$310\text{K} - \$320\text{K} = -10\text{K}$
  - We repeat the above steps by randomly drawing values of the features and take the average of the obtained contributions of the 'cat-banned' feature
- Similarly, we repeat the calculations for all possible coalitions of features
  - There are 8 possible coalitions of 'park-nearby', 'size-50', and '2nd-floor'
  - For each coalition, we compute the predicted apartment price with and without the 'cat-banned' feature, and take the difference to obtain the marginal contribution
  - The Shapley value $\phi_{\text{cat}-\text{banned}}$ will be the average of the marginal contributions $\Delta_{\pi}^{f}(\text{cat} - \text{banned})$ by the 'cat-banned' feature to all coalitions

Shapley Values

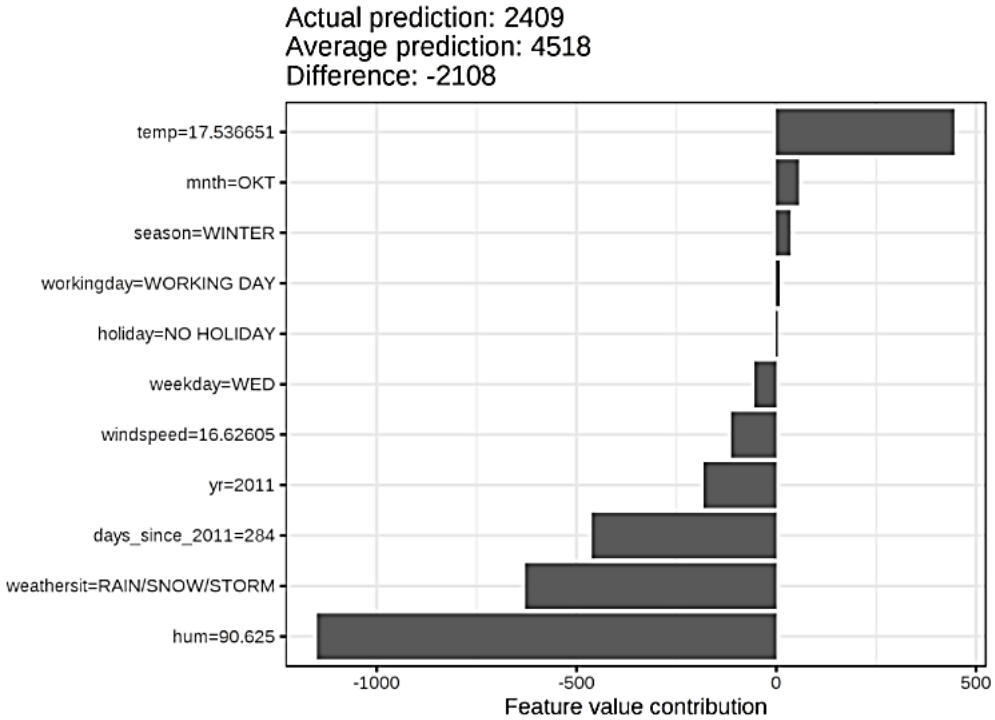
- The interpretation of the Shapley value for feature j is:
  - The j-th feature contributed ϕ_j to the prediction of this particular instance compared to the average prediction for the dataset
- For instance, the figure presents the Shapley values of different features for one person, used in a random forest model for predicting cervical cancer
  - The figure indicates that the person has high risk of cancer (0.54 probability above the average prediction of 0.03)



Picture from: https://christophm.github.io/interpretable-ml-book/shapley.html

Shapley Values

- The figure presents the Shapley values of the features for the day 285, used in a random forest model for predicting the number of rented bikes per day
  - The predicted number of rented bikes for that day is 2,409, which is -2,108 below the average daily prediction of 4,518 bikes
  - The main contributors for the lower number are the weather situation and humidity
  - Note that Shapley values can be negative



Picture from: https://christophm.github.io/interpretable-ml-book/shapley.html

SHAP

- ***SHAP (SHapley Additive exPlanations)***
  - Lundberg (2016) - A Unified Approach to Interpreting Model Predictions
- SHAP connects Shapley Values and LIME approaches, by reformulating Shapley Values and adding new properties
  - The properties improved the performance and showed better consistency with human interpretation
- SHAP defines the explanation for an instance $\mathbf{x}$ as

$$f(\mathbf{x}) = \phi_0 + \sum_{i=1}^{M} \phi_i(f, \mathbf{x})$$

  - $\phi_i(f, \mathbf{x})$ is the contribution of feature i to the explanation $f(\mathbf{x})$, and it is called the SHAP value of i
    - It is calculated based on the Shapley value: $\phi_i(f, \mathbf{x}) = \frac{1}{M!}\sum_{\pi \in \Pi}\left[f_{\mathbf{x}}(S_\pi^i \cup \{i\}) - f_{\mathbf{x}}(S_\pi^i)\right]$
  - ϕ_0 is the base value, calculated as the value of $f(\mathbf{x})$ when no feature is present, i.e., $f_{\mathbf{x}}(\emptyset)$
  - M is the number of features

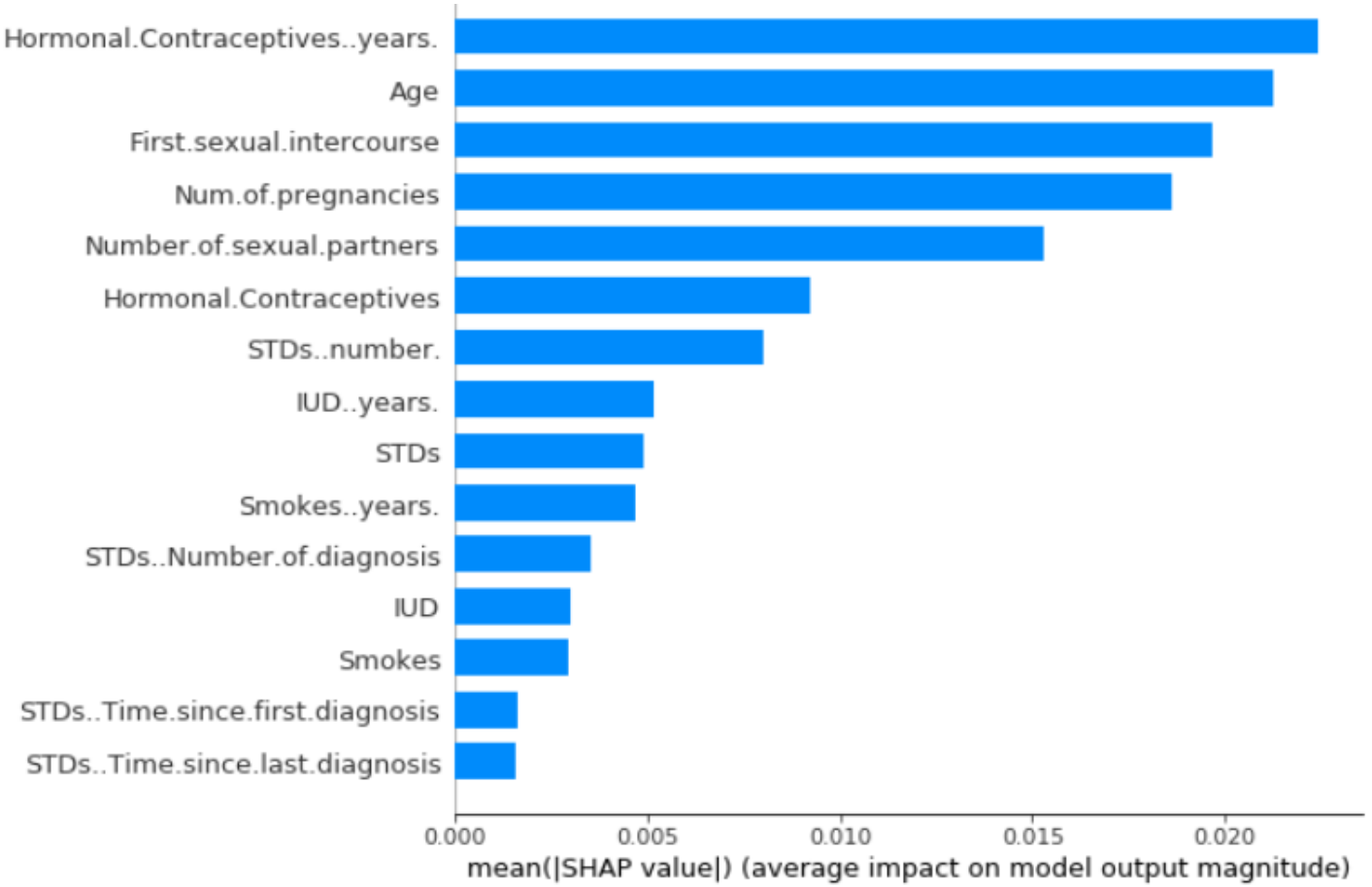
SHAP

- To calculate SHAP value, the authors propose Kernel SHAP, which combined Shapley Values and LIME
  - I.e., the black-box model $f(\mathbf{x})$ is locally approximated with a linear model $g(\boldsymbol{z})$
  - $\mathbf{z}$ is a vector that represents the presence of absence of features, i.e., $\boldsymbol{z} = (z_1, \ldots, z_M)$
  - The explanation is approximated by: $f(\mathbf{x}) \approx g(\mathbf{z}) = \phi_0 + \sum_{i=1}^{M} \phi_i(f, \mathbf{x}) z_i$
- Recall the formulation of LIME: $\arg\min_{g \in G} \mathcal{L}(f, g, \pi_{\mathrm{x}}) + \Omega(g)$
- SHAP values $\phi_i(f, \mathbf{x})$ are calculated by solving LIME using the following:
  - $\Omega(g) = 0$
  - $\pi_{\mathrm{x}}(\mathbf{z}) = \frac{M-1}{\binom{M}{|z|} |z|(M-|z|)}$
  - $\mathcal{L}(f, g, \pi_{\mathrm{x}}) = \sum_z [f(\mathbf{x}') - g(\mathbf{z})]^2 \, \pi_{\mathrm{x}}(\mathbf{z})$

  where $|z|$ is the number of non-zero elements in $\mathbf{z}$
- The authors also proposed ***TreeSHAP***, which is a variant of SHAP for tree-based models (decision trees, random forests)

SHAP

- Example: mean SHAP values per feature, using a random forest model for predicting cervical cancer
  - The number of years with hormonal contraceptives was the most important feature



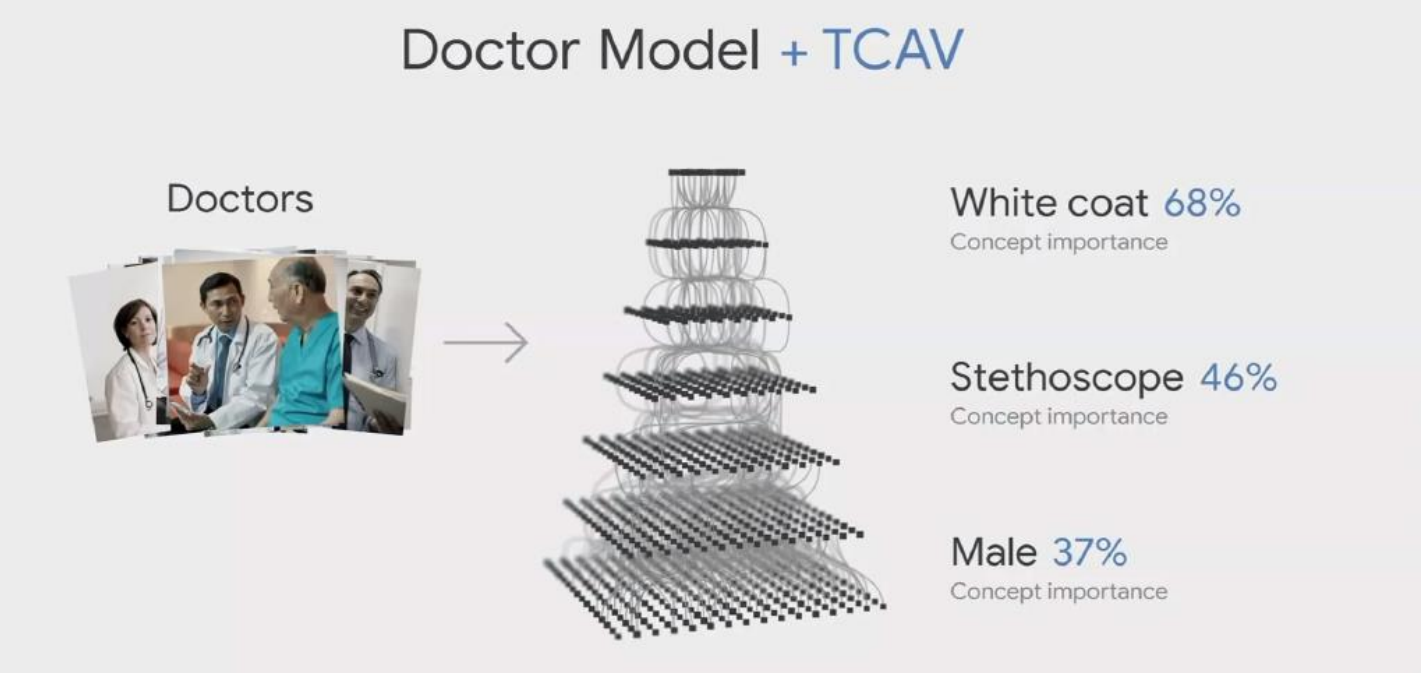
Picture from: https://christophm.github.io/interpretable-ml-book/shap.html

Type of Explanations

- Pixel-level explanations
- Feature-level explanations
- **Concept-level explanations**
  - TCAV, ACE
- Instance-level explanations

Concept-level Explanations

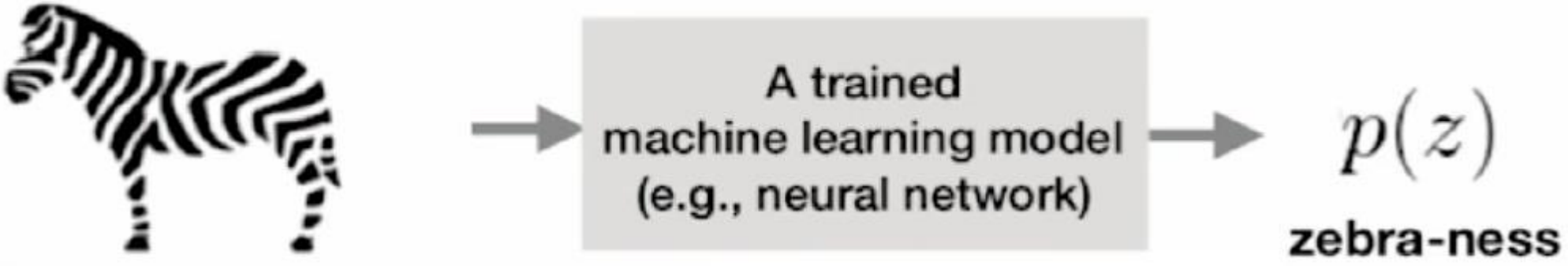
- ***Pixel-level importance***: which input pixels are important when a model classifies one input example? (Local Explanation)
- ***Concept-level importance***: which high-level concepts are important when a model classifies one class of inputs across the entire dataset? (Global Explanation)
  - E.g., the figure shows concepts for recognizing the class "doctor" in images: the training set does not have class labels for the concepts "White coat" or "Stethoscope"



Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

TCAV

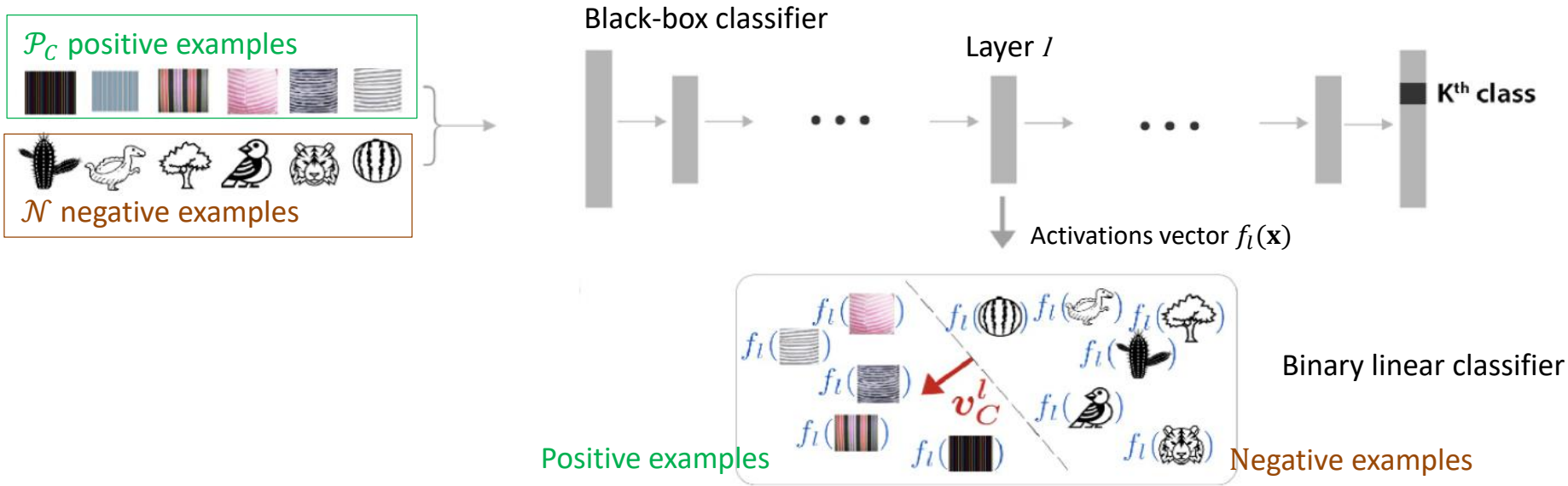
- *TCAV (Testing with Concept Activation Vectors)*
  - Kim (2018) - Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)
- For instance: how important is the concept "striped" for assigning the zebra class label by a classifier



- The approach consists of three steps:
  1. Represent the concept as a vector (referred to as Concept Activation Vector (CAV))
  2. Quantify the sensitivity of an image to the CAV vector
  3. Measure the importance of the concept to multiple images of the same class

TCAV

- Step 1: Represent the concept (e.g., "striped") as a vector
  - Collect a set of positive examples $\mathcal{P}_C$ of the concept C, and a set of negative examples $\mathcal{N}$
  - Obtain the activations $f_l(\mathbf{x})$ for each training example $\mathbf{x}$ at a layer l of the model
  - Use the layer activations $f_l(\mathbf{x})$ for all images in $\mathcal{P}_C$ and $\mathcal{N}$ to train a binary linear classifier to separate the sets $\mathcal{P}_C$ and $\mathcal{N}$ into: $\{f_l(\mathbf{x}): \mathbf{x} \in \mathcal{P}_C\}$ and $\{f_l(\mathbf{x}): \mathbf{x} \in \mathcal{N}\}$
  - The vector v_c^l that is normal to the decision boundary of the linear classifier (the red arrow) is called concept activation vector (CAV) for the concept C
    - The CAV vector v_c^l represents the orthogonal direction to the decision boundary, along which the probability of the concept class C increases the fastest



TCAV

- To assess whether the CAV vector v_c^l captures the intended concept C, use it to sort images in an external set (not in the training sets $\mathcal{P}_C$ and $\mathcal{N}$)
  - E.g., the following are the top 3 and the bottom 3 images for two concepts
    - Concept "suit" for the class CEO (left), and concept "lab coat" for the class boss (right)

TCAV

- Step 2: Calculate directional derivative of the class score w.r.t concept
  - Let $\mathbf{x}$ be an image with a predicted class k, and $z_k(\mathbf{x})$ be the prediction score (logit value) by the black-box model for the class k
    - $z_k(\mathbf{x})$ is also a function of the activations $f_l(\mathbf{x})$ at layer l, and it can be written as $z_k\big(f_l(\mathbf{x})\big)$
  - Pixel-level explainable approaches typically employ the gradient $\frac{\partial z_k(\mathbf{x})}{\partial x_i}$, which measures how sensitive the class score z_k is to small perturbations to the pixel x_i
    - I.e., the gradient quantifies how important the pixel x_i is to predicting the class k
  - TCAV introduces directional derivative (denoted $S_{C,k,l}(\mathbf{x})$) to measure how sensitive the class score z_k is to small perturbations in the direction of the CAV vector v_c^l :

$$S_{C,k,l}(\mathbf{x}) = \lim_{\epsilon \to 0} \frac{z_k\big(f_l(\mathbf{x}) + \epsilon \cdot v_c^l\big) - z_k\big(f_l(\mathbf{x})\big)}{\epsilon} = \nabla z_k\big(f_l(\mathbf{x})\big) \cdot v_c^l$$

  - E.g., for the shown image of zebra $\mathbf{x}$, the directional derviative $S_{C,k,l}(\mathbf{x})$ quantifies the sensitivity of the class k logit in the black-box classifier to the concept "striped" at the layer l

$$S_{C,k,l}(\text{  }) = \nabla z_k\,(f_l(\text{  })) \cdot \boldsymbol{v}_C^l$$

TCAV

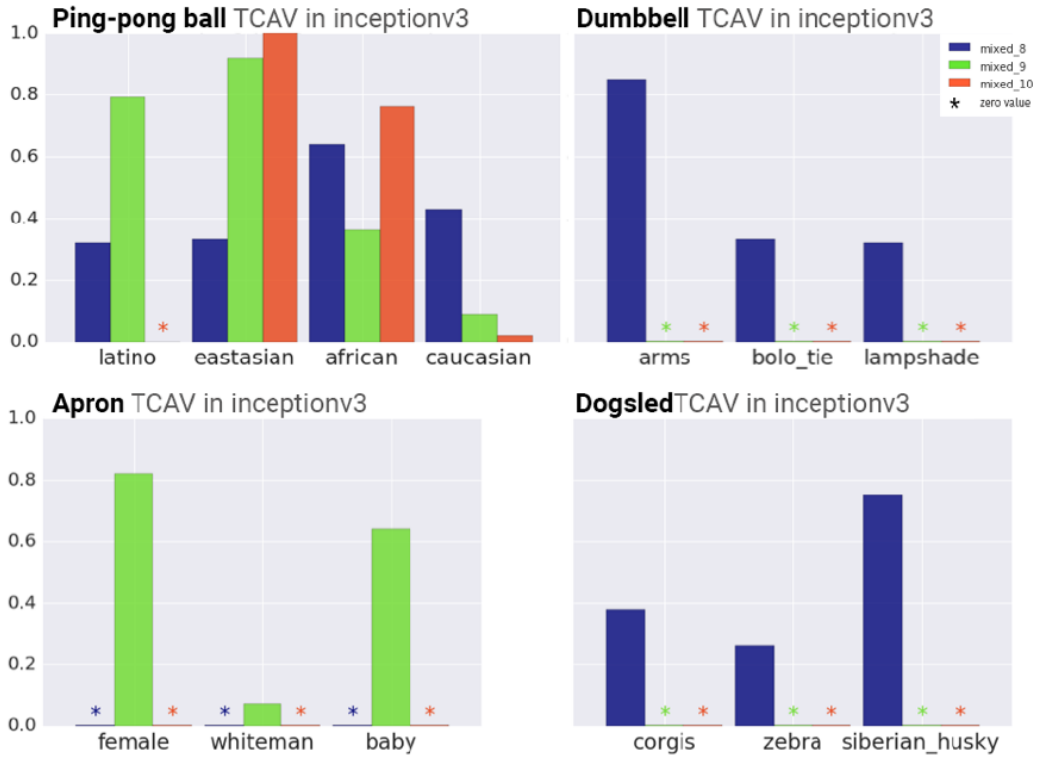
- Step 3: Testing with CAVs (TCAV)
  - For the set of all input instances $\mathbf{X}_k$ with the ground truth label k, TCAV is the fraction of instances whose l-layer CAV vector has positive value

$$TCAV(C, k\ l) = \frac{\left|\{\mathbf{x} \in \mathbf{X}_k : S_{C,k,l}(\mathbf{x}) > 0\}\right|}{|\mathbf{X}_k|}$$

  - $TCAV$ measures how important the concept C is to the model for assigning the class label k to the set of input instances $\mathbf{X}_k$
    - I.e., it is global metric of the concept C for all images with a given label
    - Note that $TCAV(C, k\ l) \in [0, 1]$
  - To eliminate spurious correlations, the authors calculated TCAV scores multiple times with different random sets of images, and applied a t-test hypothesis testing
    - If $\mathcal{P}_C$ and $\mathcal{N}$ do not have significant impact on obtained TCAV values, the TCAVs for the positive and negative examples would be equal (i.e., 0.5)
- The major drawback of the TCAV approach:
  - Requires to manually construct positive sets of examples $\mathcal{P}_C$ that contain the concept of interest
    - The negative set $\mathcal{N}$ can include any random images that don't contain the concept

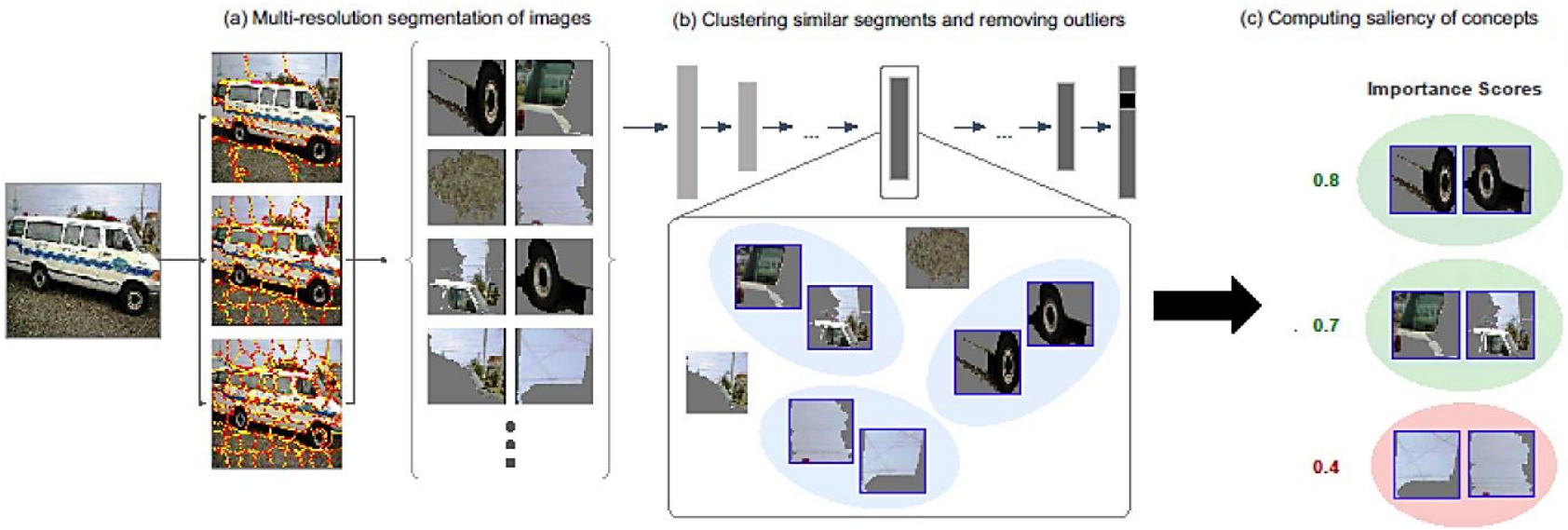
TCAV

- TCAV evaluation of the Inception v3 model for different concepts
  - E.g., in the upper-left figure, TCAVs are shown for the concepts: Latino, East Asian, African, and Caucasian, with respect to the class "ping-pong ball"
    - The bars show the TCAVs for 3 different layers of Inception v3 (mixed 8, 9, and 10)
    - None of these concepts are included in the ground-truth class labels in the dataset
    - There is a high correlation between the concept East Asian and the class ping-pong ball



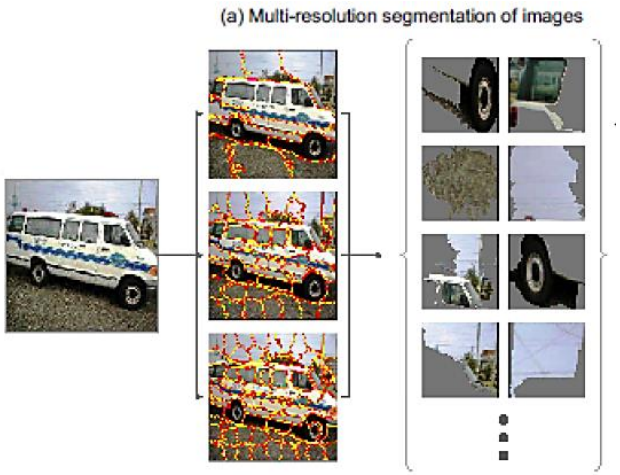
ACE

- *ACE (Automatic Concept-based Explanations)*
  - Ghorbani (2019) - Towards Automatic Concept-based Explanations
- ACE automatically extracts visual concepts
  a) Images from the same class are segmented at multiple resolutions, resulting in a pool of image patches
  b) The obtained image segments are resized to the input resolution, and activation vectors from one layer of the black-box model are used to cluster similar segments
  c) TCAV scores are used to sort the clusters of concepts based on their importance

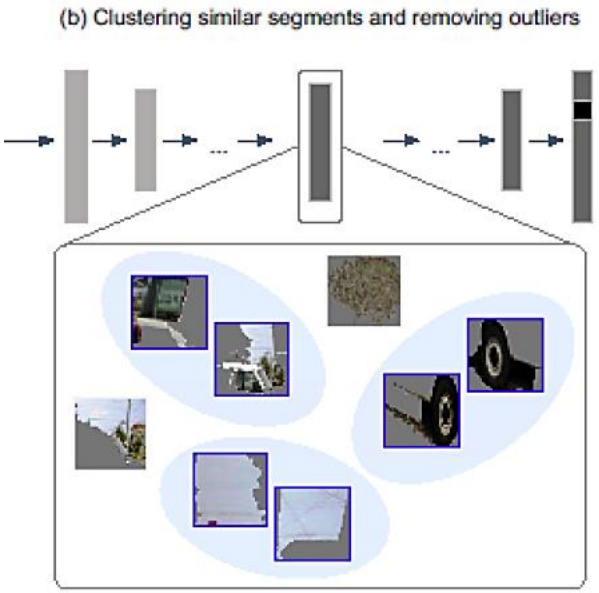


ACE

- The multiple resolutions of the extracted image segments extract concepts at both fine-grained and course-grained abstractions (e.g., textures, parts, objects)
  - The authors use three different resolutions of segments
  - An existing approach for image segmentation based on super-pixels was applied (middle image on the right)



- Similar segments are grouped together using k-mean clustering based on the Euclidean (ℓ_2) distance between the activation vectors from one layer of the back-box model
  - Outlier segments that have low similarity to the clustered segments were removed
- ACE advantage over the TCAV approach:
  - It does not require a manual selection of a set of positive and negative images for a concept



ACE

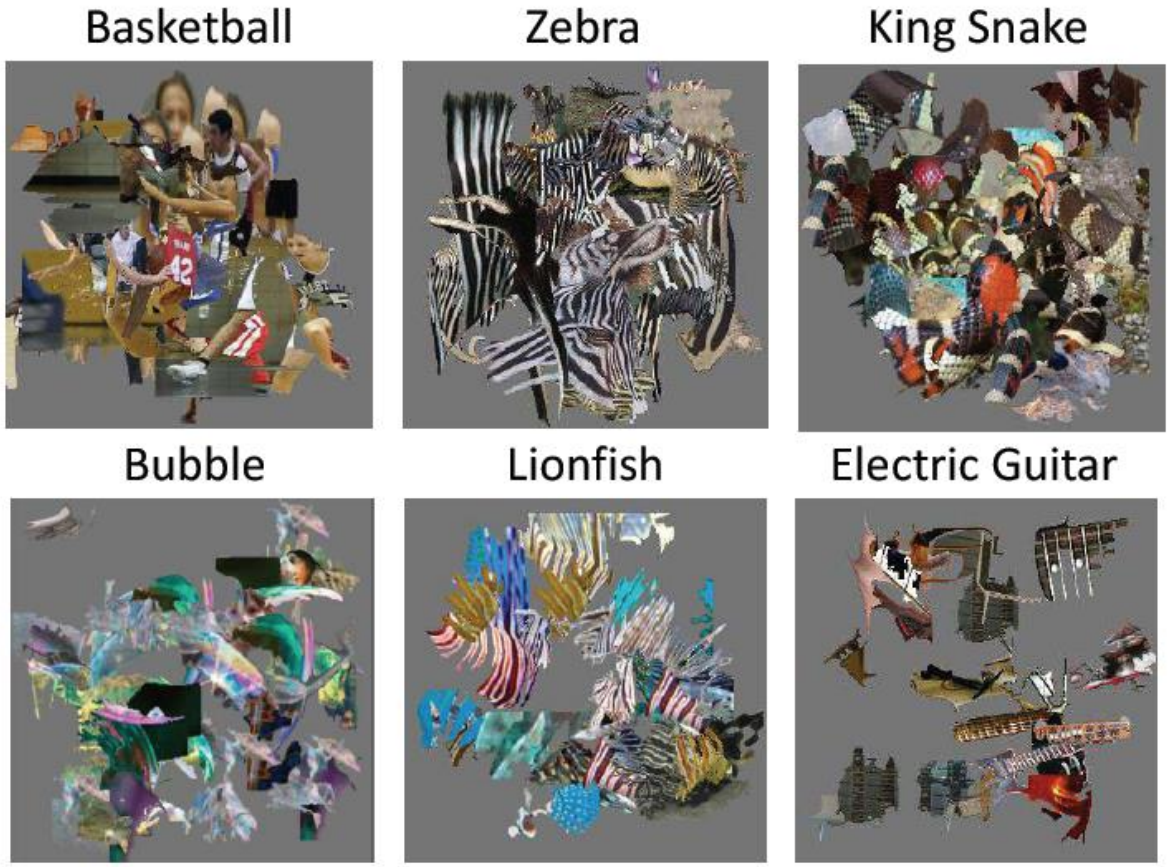
- Experimental evaluation was performed on the Inception v3 model by selecting the activations from the mixed_8 layer for concept explanation
- E.g., for the "Police Van" class, the four most important concepts are: (1) tires, (2) letters of the Police logo, (3) windows, and (4) fenders

ACE

- Similarly, for the "basketball" class, the four most important concepts are: (1) player's jersey, (2) ball, (3) court's floor, and (4) player's hand
  - Interestingly, the player's jersey is a more important concept than the ball

ACE

- In addition, the trained Inception v3 model can make the correct class prediction on images with randomly stitched concept segments for that class
  - That is, CNNs for image classification overly rely on pattern (texture) recognition
    - Meaningless combinations of patterns from a class result in correct class prediction



Type of Explanations

- Pixel-level explanations
- Feature-level explanations
- Concept-level explanations
- **Instance-level explanations**
  - Prototypes and criticisms, counterfactual explanations

Instance-level Explanations

- ***Instance-based explanation*** methods use particular input instances to explain the behavior of a black-box model
- E.g., *k*-nearest neighbors is an interpretable model which employs *k* data instances to make a prediction
  - The *k*-NN prediction for one input instance can be explained based on the characteristics of the *k* neighbors
- Similar, we use past experiences to make new predictions
  - E.g., a doctor seeing a patient with specific symptoms may remind her of another patient with the same symptoms, and may suspect that the patient may have the same disease
  - E.g., an employee who works on risk analysis may recall a similar project that he completed in the past, and he may decide to reuse the same approach for the current task
- Instance-based explainability approaches are based on:
  - Prototypes or criticisms
  - Counterfactual examples

Prototypes and Criticisms

- ***Prototypes and criticisms***
  - Kim (2016) - Examples are not Enough, Learn to Criticize! Criticism for Interpretability
- It is a global explanation method that explains class predictions by a black-box model based on collections of representative instances
  - ***Prototypes***: representative examples of the class
  - ***Criticisms***: examples of the class that are not well represented by the prototypes
- The authors used Maximum Mean Discrepancy (MMD) to measure the discrepancy between two distributions, given by

$$MMD^2 = \frac{1}{m^2}\sum k(z_i, z_j) - \frac{2}{mn}\sum k(z_i, x_j) + \frac{1}{n^2}\sum k(x_i, x_j)$$

  - m is the number of prototypes z, n is the number of instances x of the original dataset
  - k is a kernel function that measures the similarity of two instances, which is adopted as the radial basis function $k(x, x') = exp(-\gamma\|x - x'\|^2)$, with a scaling parameter γ
- The approach is referred to as MMD-critic
  - Prototypes are selected so that their distribution is close to the input data distribution, whereas the distribution of criticisms is far from the data distribution

Prototypes and Criticisms

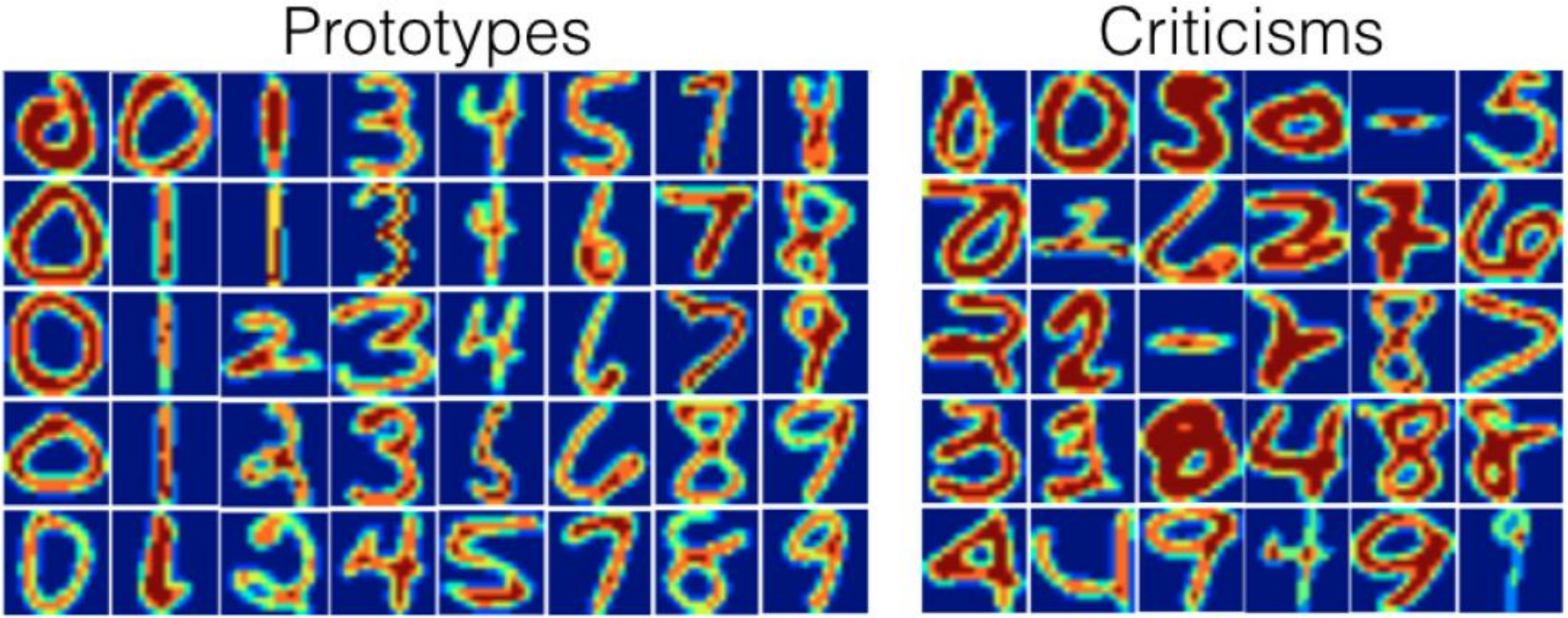
- To find prototypes:
  1. Start with an empty set of prototypes
  2. While the number of prototypes is below the chosen number m:
     - For each instance in the dataset, check how much MMD is reduced when the instance is added to the set of prototypes
     - Add the data instance that minimizes MMD to the set of prototypes
  3. Return the set of prototypes
- To find criticisms, the following witness function is used:

$$\text{witness}(x) = \frac{1}{n}\sum k(x, x_i) - \frac{1}{m}\sum k(x, z_j)$$

  - The witness function evaluates which of two distributions fits the instance x better
- For a trained black-box model, the predicted classes for the prototypes and criticisms can help to understand the model
  - E.g., via analysis of the cases when the model made wrong predictions

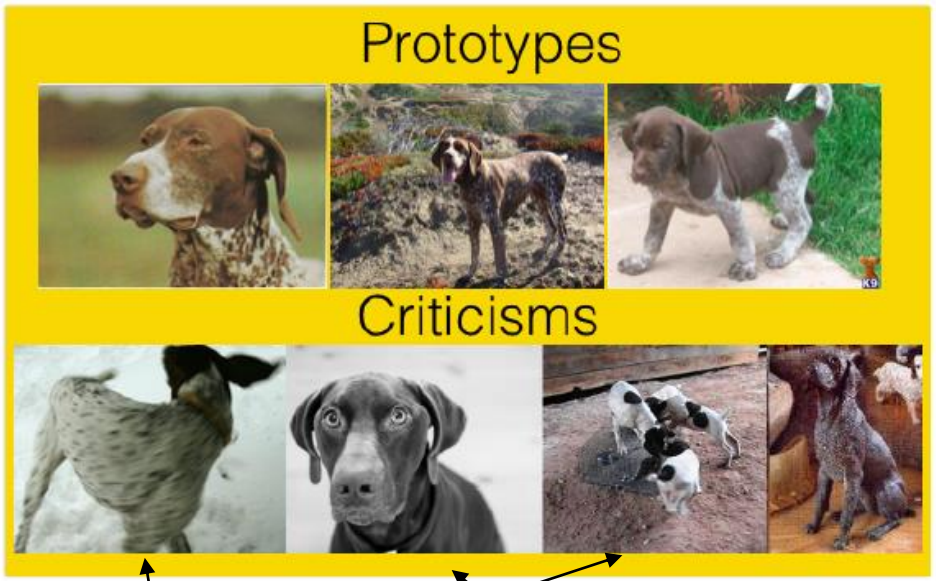
Prototypes and Criticisms

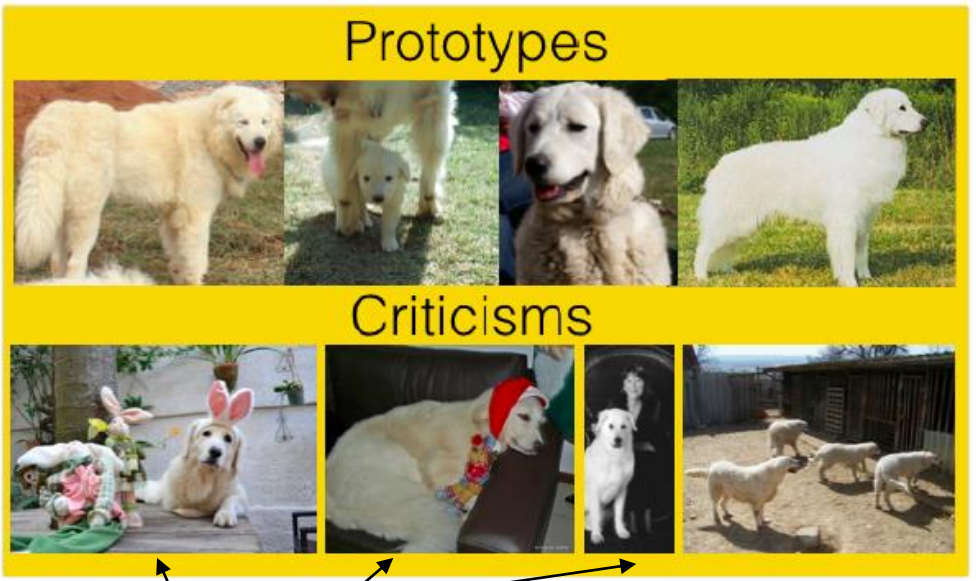
- The figure shows prototypes and criticisms from the USPS dataset of handwritten digits
  - The prototypes represent common way of writing digits, whereas the criticisms represent outliers and ambiguously written digits



Prototypes and Criticisms

- Protypes and criticisms for two breeds of dogs from the ImageNet dataset
- The addition of criticisms examples to the prototypes made it easier for human subjects to identify the defining features of different classes and improved the interpretability





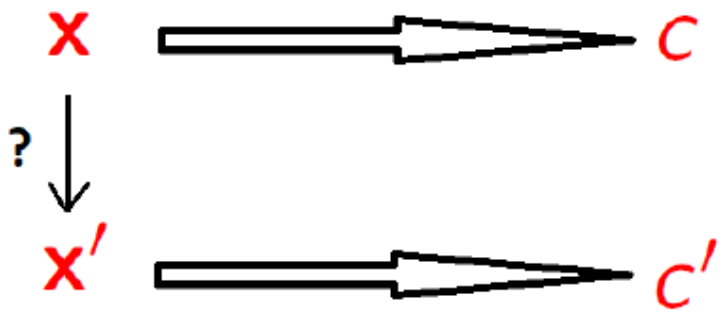
Criticism: dogs in movement

Criticism: black-and-white image

Criticism: dogs in costumes

Counterfactual Explanations

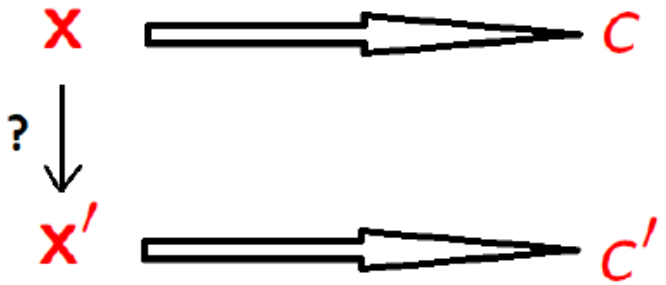
- *Counterfactuals*: Help to answer "what would have happened if" question
  - Wachter (2017) – Counterfactual Explanations Without Opening the Black Box: Automated Decision and the GDPR
- Consider the case where an ML model was used for deciding on a loan application
  - E.g., John's feature vector is x, and his loan application was denied by the model (it was assigned to the class c)
- **Counterfactual explanation**
  - The loan would have been granted if income was $45,000 instead of $30,000
  - That is, the application would have been assigned to class c' instead of c, if John's feature vector was x' instead of x



Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

Counterfactual Explanations

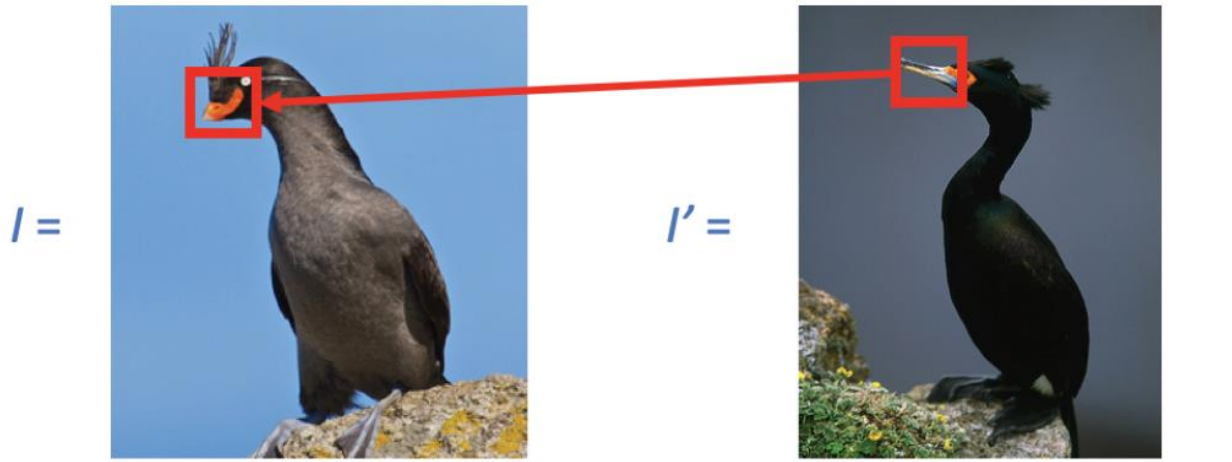
- Finding a counterfactual example x' to an input example x:
  - The difference between the counterfactual example x' and the original input example x should be small
  - The counterfactual example x' should be misclassified by the model as another class c' (whereas the original input example x is classified as class c)
  - The change from x to x' should be feasible in the real-world (actionability)
    - Actionable examples: the loan would be granted if income was \$45,000 instead of \$30,000
    - Not actionable example: the loan would be granted if age was 30 instead of 50
- Note that the above procedure is the same as when creating adversarial examples in ML
  - Adversarial examples are also counterfactuals, only they are used to attack the model, rather than to explain it



Slide credit: Nevin L. Zhang – Lecture 9: Explainable AI

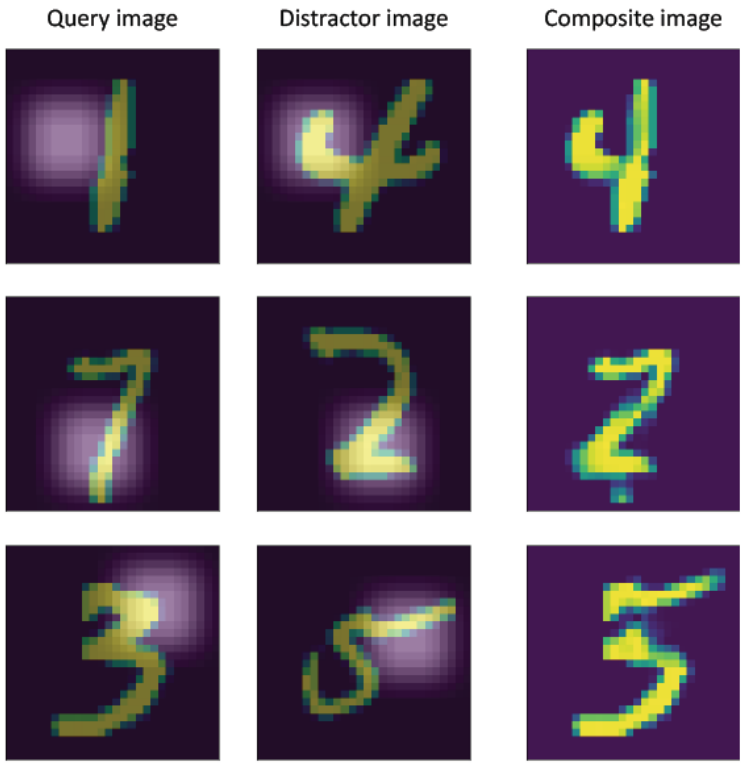
Counterfactual Explanations

- ***Counterfactual visual explanations***
  - Goyal (2019) - Counterfactual Visual Explanations
- Find a counterfactual visual example for the left image *I* classified by the model as class *c* (*Crested Auklet*), so that the model would classify it as class *c′* (*Red Faced Cormorant*)
  - Select a distractor image *I′* (right figure) of the class c′
  - Identify regions in *I* and *I′* (outlined with the red squares) such that if the region in *I* is replaced with the region in *I′*, the model would classify *I* as c′
  - Counterfactuals explain the most important features for classifying an image



Counterfactual Explanations

- Example: the left images are original input images of a class (digits 1, 7, or 3)
  - The middle column has a distractor image for misclassifying the original image
    - The highlighted pixels identify the regions for misclassifying the original image
    - E.g., if the highlighted region in 1 is replaced with the highlighted region in 4, the class of the image of digit 1 would change to 4
  - The composite images show the original images with the highlighted region of the distractor images added



Additional References

1. Nevin L. Zhang (Hong Kong University of Science and Technology) – Machine Learning course, Lecture 9 – Explainable AI
2. Belle et al. (2020) – Principles and Practice of Explainable Machine Learning
3. Christoph Molnar (2020) – Interpretable Machine Learning: A Guide for Making Black Box Models Explainable
4. Arietta et al. (2019) – Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI
5. Geyik, KDD 2019 Tutorial, Explainable AI in Industry, 2019.
6. Hima Lakkaraju, CS282BR: Topics in Machine Learning. Interpretability and Explainability, 2020.
7. Freddy Lecue, XAI – Explanation in AI: From Machine Learning to Knowledge Representation & Reasoning and Beyond, 2019.