CENG7880
Trustworthy and Responsible AI

Instructor: Sinan Kalkan

(https://ceng.metu.edu.tr/~skalkan)

For course logistics and materials:
https://metu-trai.github.io

MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Previously on CENG7880

Aleatoric vs. Epistemic Uncertainty

- The **epistemic uncertainty** is

$$\text{Epistemic}(x) = f_{\beta^*}(x) - f_{\hat{\beta}(Z)}(x)$$

  - Here, $\text{Epistemic}(x)$ is a random function of the random variable $Z \sim p^n$
  - Thus, $\text{Epistemic}(x)$ is itself a random variable

- **Goal:** Estimate the distribution of $\text{Epistemic}(x)$

  But we don't know β^*

- **Assumption:** Our model is **unbiased**: $\mathbb{E}_Z\left[f_{\hat{\beta}(Z)}(x)\right] = f_{\beta^*}(x)$

Previously on CENG7880

Typo here with the sign since the definition is the other way around: $\text{Epistemic}(x) = f_{\beta^*}(x) - f_{\hat{\beta}(Z)}(x)$

- $\{\hat{f}_i(x) - f_{\beta^*}(x)\}_{i=1}^{k}$ are i.i.d. samples from $\text{Epistemic}(x)$
- By our unbiasedness assumption:

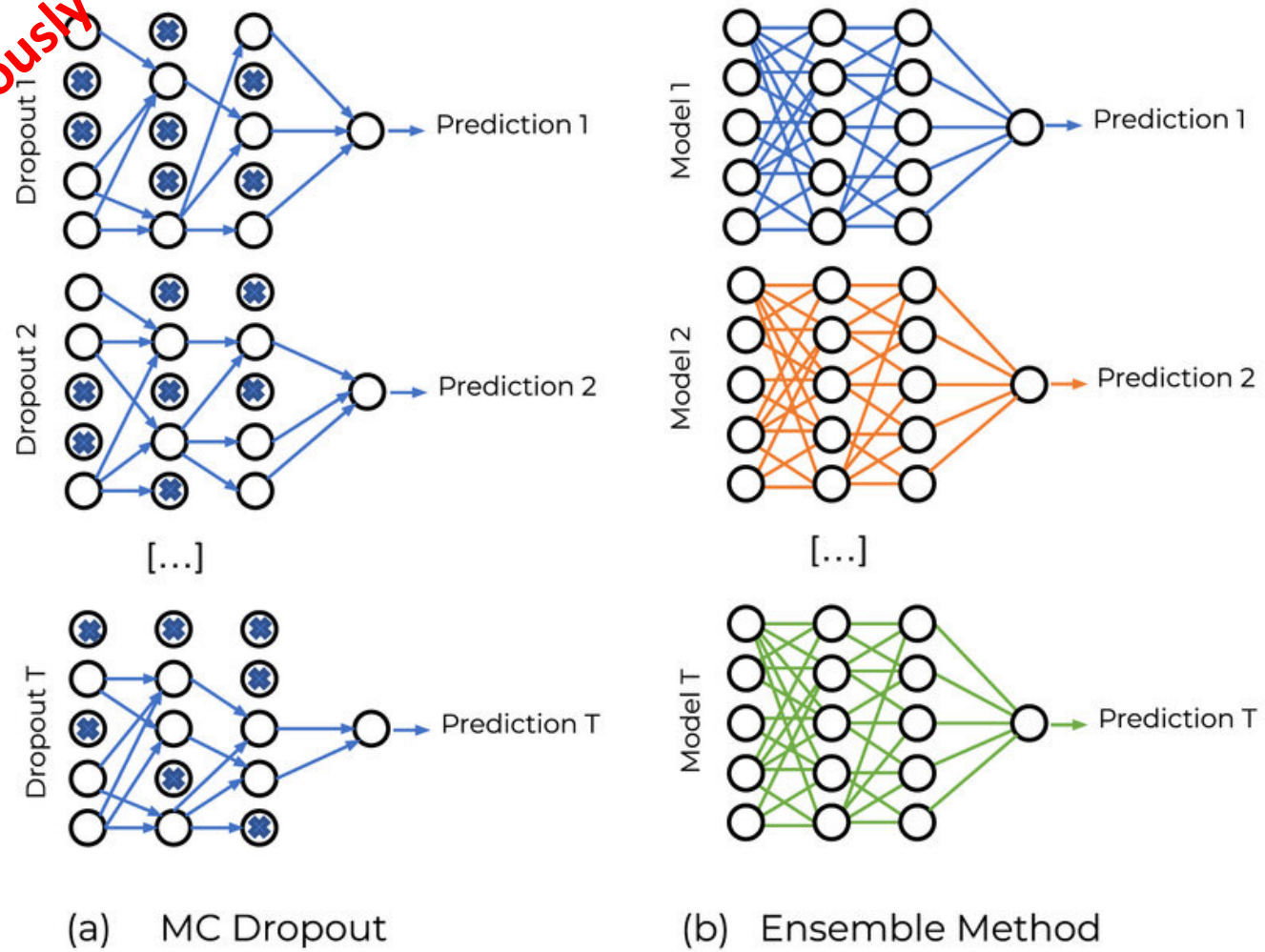
$$f_{\beta^*}(x) = \mathbb{E}_Z\left[f_{\hat{\beta}(Z)}(x)\right] \approx k^{-1}\sum_{i=1}^{k}\hat{f}_i(x) := \hat{\mu}(x)$$

Same architecture trained on different subsets/versions of Z.

- $\{\hat{f}_i(x) - \hat{\mu}(x)\}_{i=1}^{k}$ are approximately i.i.d. samples from $\text{Epistemic}(x)$
  - **Problem:** We cannot take unlimited samples from P
  - Only have a single training dataset Z!

Previously on CENG7880

Method for Estimating Epistemic Uncertainty



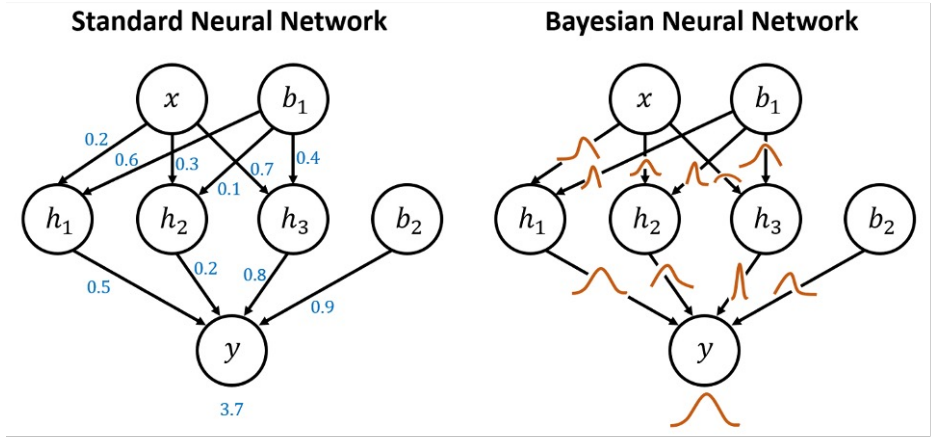


Fig: https://medium.com/@costaleirbag/a-first-insight-into-bayesian-neural-networks-bnn-c767551e9526

Fig: https://www.researchgate.net/figure/a-MC-Dropout-at-inference-time-The-T-predictions-are-obtained-from-Dropout-of_fig10_359970124

Previously on CENG7880

Method for Estimating Epistemic Uncertainty: Monte Carlo Dropout

For classification problems:

For classification this can be approximated using Monte Carlo integration as follows:

$$p(y = c|\mathbf{x}, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T} \sum_{t=1}^{T} \text{Softmax}(\mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})) \tag{3}$$

with T sampled masked model weights $\widehat{\mathbf{W}}_t \sim q_\theta^*(\mathbf{W})$, where $q_\theta(\mathbf{W})$ is the Dropout distribution [6]. The uncertainty of this probability vector $\mathbf{p}$ can then be summarised using the entropy of the probability vector: $H(\mathbf{p}) = -\sum_{c=1}^{C} p_c \log p_c$. For regression this epistemic uncertainty is captured

"What uncertainties do we need for computer vision?"
https://arxiv.org/pdf/1703.04977

Previously on CENG7880

Method for Estimating Epistemic Uncertainty: Monte Carlo Dropout

For regression problems:

probability vector: $H(\mathbf{p}) = -\sum_{c=1}^{C} p_c \log p_c$. For regression this epistemic uncertainty is captured by the predictive variance, which can be approximated as:

$$\text{Var}(\mathbf{y}) \approx \sigma^2 + \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})^T \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t) - E(\mathbf{y})^T E(\mathbf{y}) \quad (4)$$

with predictions in this epistemic model done by approximating the predictive mean: $E(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})$. The first term in the predictive variance, σ^2, corresponds to the amount of noise inherent in the data (which will be explained in more detail soon). The second part of the predictive variance measures how much the model is uncertain about its predictions – this term will vanish when we have zero parameter uncertainty (i.e. when all draws $\widehat{\mathbf{W}}_t$ take the same constant value).

"What uncertainties do we need for computer vision?"
https://arxiv.org/pdf/1703.04977

Previously on CENG7880

Methods for Estimating Aleatoric Uncertainty

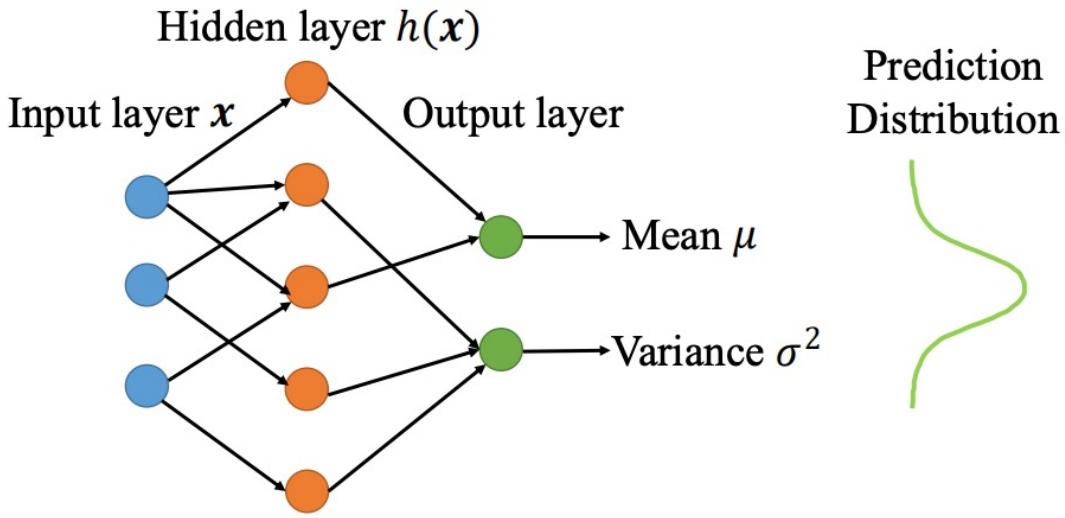
For regression problems:

- Aleatoric uncertainty is modeled by the distribution:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta})$$

- In a regression problem, we can obtain this as:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma_{\boldsymbol{\theta}}(\boldsymbol{x}))$$



(a) Prediction distribution example.

$$\mathcal{L}_{\mathrm{NN}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2\sigma_{\boldsymbol{\theta}}^2(\boldsymbol{x}_i)} ||y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)||^2 + \frac{1}{2}\log \sigma_{\boldsymbol{\theta}}^2(\boldsymbol{x}_i).$$

https://arxiv.org/pdf/2302.13425

Previously on CENG7880

Methods for Estimating Aleatoric Uncertainty

For classification problems:

(passing inputs through the model to get logits). We only need to sample from the logits, which is a fraction of the network's compute, and therefore does not significantly increase the model's test time. We can rewrite the above and obtain the following numerically-stable stochastic loss:

$$\hat{\mathbf{x}}_{i,t} = \mathbf{f}_i^{\mathbf{W}} + \sigma_i^{\mathbf{W}} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

$$\mathcal{L}_x = \sum_i \log \frac{1}{T} \sum_t \exp(\hat{x}_{i,t,c} - \log \sum_{c'} \exp \hat{x}_{i,t,c'}) \tag{12}$$

with $x_{i,t,c'}$ the c' element in the logit vector $\mathbf{x}_{i,t}$.

"What uncertainties do we need for computer vision?"
https://arxiv.org/pdf/1703.04977

Previously on CENG7880

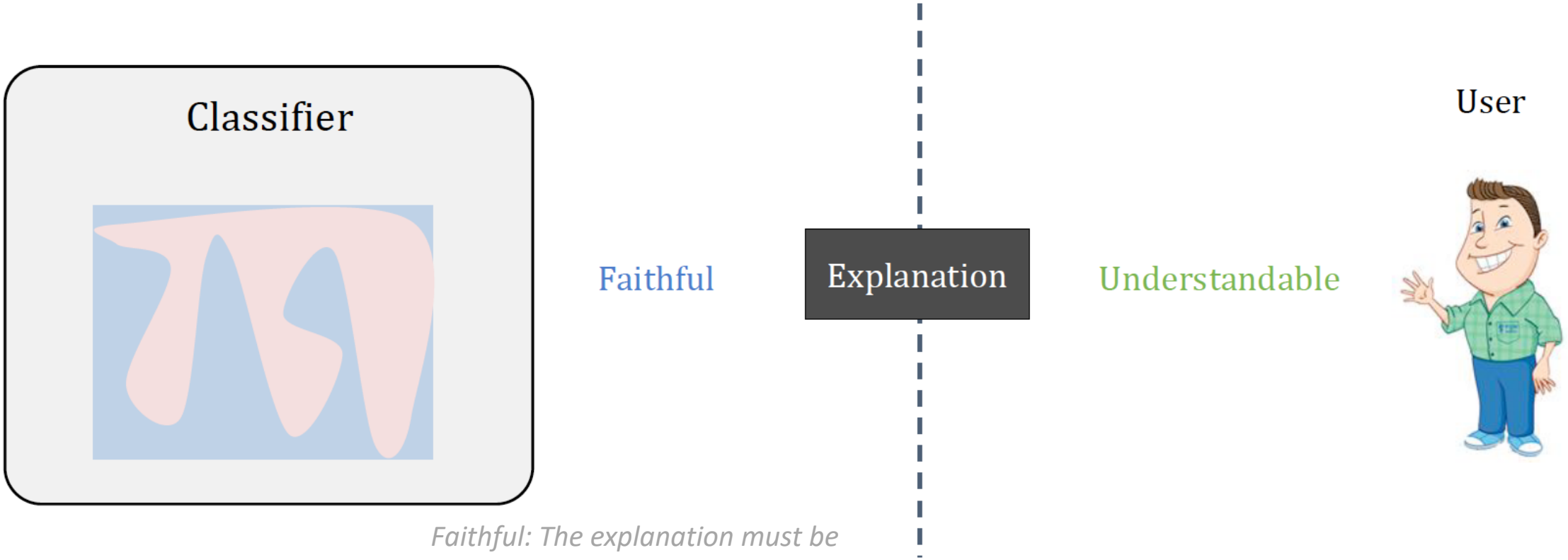
Explainable AI

- Interpretable models vs. posthoc explanations
- Local vs. global explainability
- Feature attribution methods
- Saliency methods
- Causality
- Concept Bottleneck Models

Previously on CENG7880

What is an Explanation ?

Ideally, interpretable description of the model behavior



Faithful: The explanation must be true to the model's operation

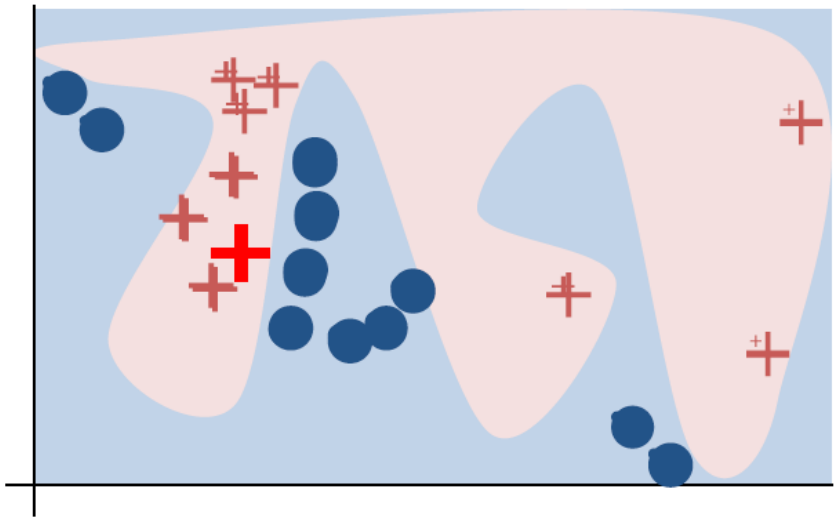
16

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Previously on CENG7880

LIME Algorithm

1. Sample points around x
2. Use model to predict labels for each sample
3. Weigh samples according to distance to x



$$\pi_x(z) = \exp\left(-\frac{D(x,z)^2}{\sigma^2}\right)$$

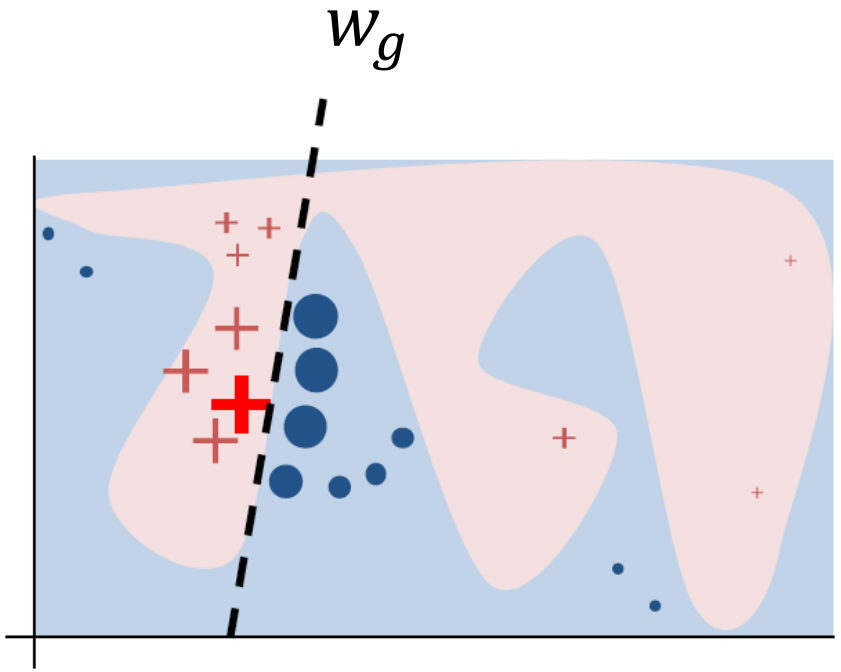
where $D(x,z)$ can be e.g., cosine distance for text and L2-distance for images.

28

Previously on CENG7880

LIME Algorithm

1. Sample points around x
2. Use model to predict labels for each sample
3. Weigh samples according to distance to x
4. Learn simple linear model on weighted samples



$$g(z') = w_g \cdot z'$$

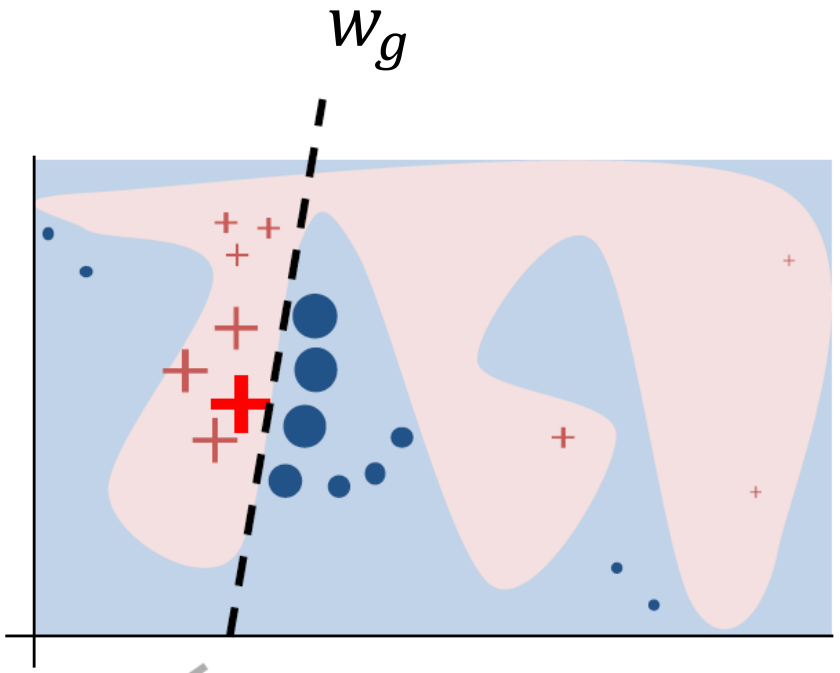
$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

29

Previously on CENG7880

LIME Algorithm

1. Sample points around x
2. Use model to predict labels for each sample
3. Weigh samples according to distance to x
4. Learn simple linear model on weighted samples
5. Use simple linear model to explain



$$g(z') = w_g \cdot z'$$

w_g of this model includes one coefficient (importance) for each feature.

30

Finding Explanable Model

Explainable model g:

Choose linear models here

Dataset:

Z (contains data & label & additional distance metric)

Objective function:

Local fidelity Interpretability

$$\min \quad \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

Ω(g): A measure of complexity (as opposed to interpretability)

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

Explanation:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \quad \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

(Corresponding weight for each feature)

Raw Input:

"You are a very nice person" (Label: 1)

Perturbed sample:

Interpretable binary vector indicating the presence or absence of a word (Feed into the explainable model)

z1': 0 1 1 0 0 0

z2': 0 0 1 1 1 0

z3: 1 0 0 1 0 0

z4': 1 1 1 1 0 0

z5': 0 1 0 1 1 0

Z ← {}

Z ← Z U (z1', f(z1), πx(z1)).

Z ← Z U (z2', f(z2), πx(z2)).

Z ← Z U (z3', f(z3), πx(z3)).

Z ← Z U (z4', f(z4), πx(z4)).

Z ← Z U (z5', f(z5), πx(z5)).

37

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

LIME & Conformal Prediction tutorial

- LIME:

  https://colab.research.google.com/drive/1z_2FIUVC6x3EbYTwOaIyLRLlNBhaxL7d?usp=sharing

- Conformal Prediction:

  https://colab.research.google.com/drive/1o96-AFtr_LwjuV2hW73PFNmOkOqTfvTD?usp=sharing

Previously on CENG7880

Shapley Values: Introduction and Intuition

- Relies on Shapley values
  - "In cooperative game theory, the Shapley value is a method (solution concept) for fairly distributing the total gains or costs among a group of players who have collaborated." – Wikipedia
  - Players form coalitions based on their strategies and get "profit" based on their contributions.

- In ML, game = prediction, features = players.
  - For a certain input $\mathbf{x}$,
    - treat each feature dimension value $\mathbf{x}_i$ as a "player"
    - prediction of the model is the "payout"
    - "fairly" distribute the payout among the features (players)

Shapley Values: Introduction and Intuition

Previously on CENG7880

- Example: Predict an apartment's price given its features:
  - Area, floor, park-proximity, cats-allowed
  - Goal: Explain the importance/contribution of each feature to the price (or in comparison to an average price)

- Potential answer:
  - park-nearby: €30,000
  - area: €10,000
  - floor-2nd: €0
  - cat-banned: - €50,000.
  - When added, these can explain the price of the apartment in comparison to other apartments

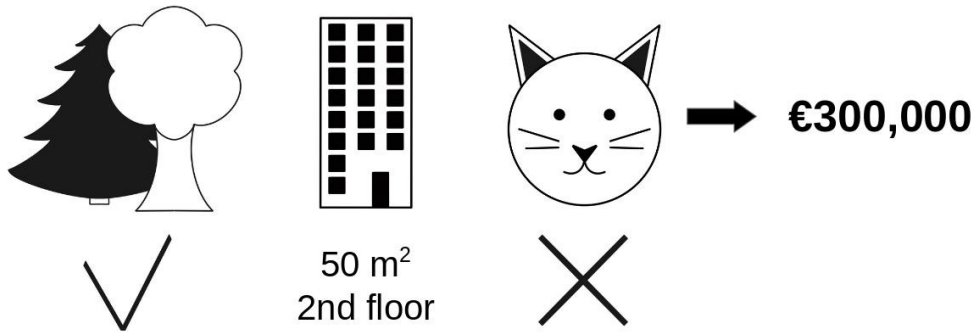


Fig & Material: https://christophm.github.io/interpretable-ml-book/shapley.html

Shapley Values: Introduction and Intuition

- Shapley value for a feature is the average contribution of a feature across all possible coalitions
- How to calculate the contribution of e.g. cat-banned?
  - Compare two coalitions with cat-banned and not cat-banned.
  - The difference is the contribution of cat-banned.
- Repeat this for more samples & coalitions.

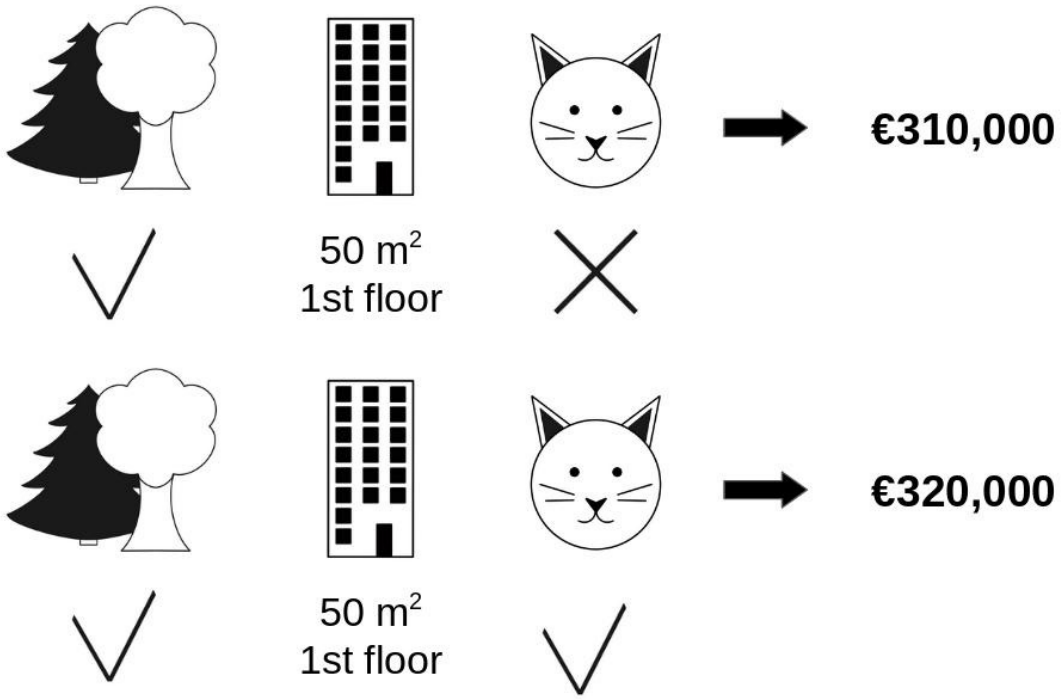


Fig & Material: https://christophm.github.io/interpretable-ml-book/shapley.html

Previously on CENG7880

Estimating Shapley Values

- Calculations with all possible coalitions can be expensive for large dimensions
- Alternative: Monte-Carlo Sampling

"Explaining Prediction Models and Individual Predictions with Feature Contributions." *2014.*

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} \left(\hat{f}(\mathbf{x}_{+j}^{(m)}) - \hat{f}(\mathbf{x}_{-j}^{(m)}) \right)$$

where $\hat{f}(\mathbf{x}_{+j}^{(m)})$ is the prediction for $\mathbf{x}$, but with a random number of feature values replaced by feature values from a random data point $\mathbf{z}$, except for the respective value of feature j. The feature vector $\mathbf{x}_{-j}^{(m)}$ is almost identical to $\mathbf{x}_{+j}^{(m)}$, but the value $x_j^{(m)}$ is also taken from the sampled $\mathbf{z}$. Each

Fig & Material: https://christophm.github.io/interpretable-ml-book/shapley.html

$$\pi_{\mathbf{x}}(\mathbf{z}') = \frac{(M-1)}{\binom{M}{|\mathbf{z}'|}|\mathbf{z}'|(M-|\mathbf{z}'|)}$$

We have the data, the target and the weights; Everything we need to build our weighted linear regression model:

$$g(\mathbf{z}') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j$$

We train the linear model g by optimizing the following loss function L:

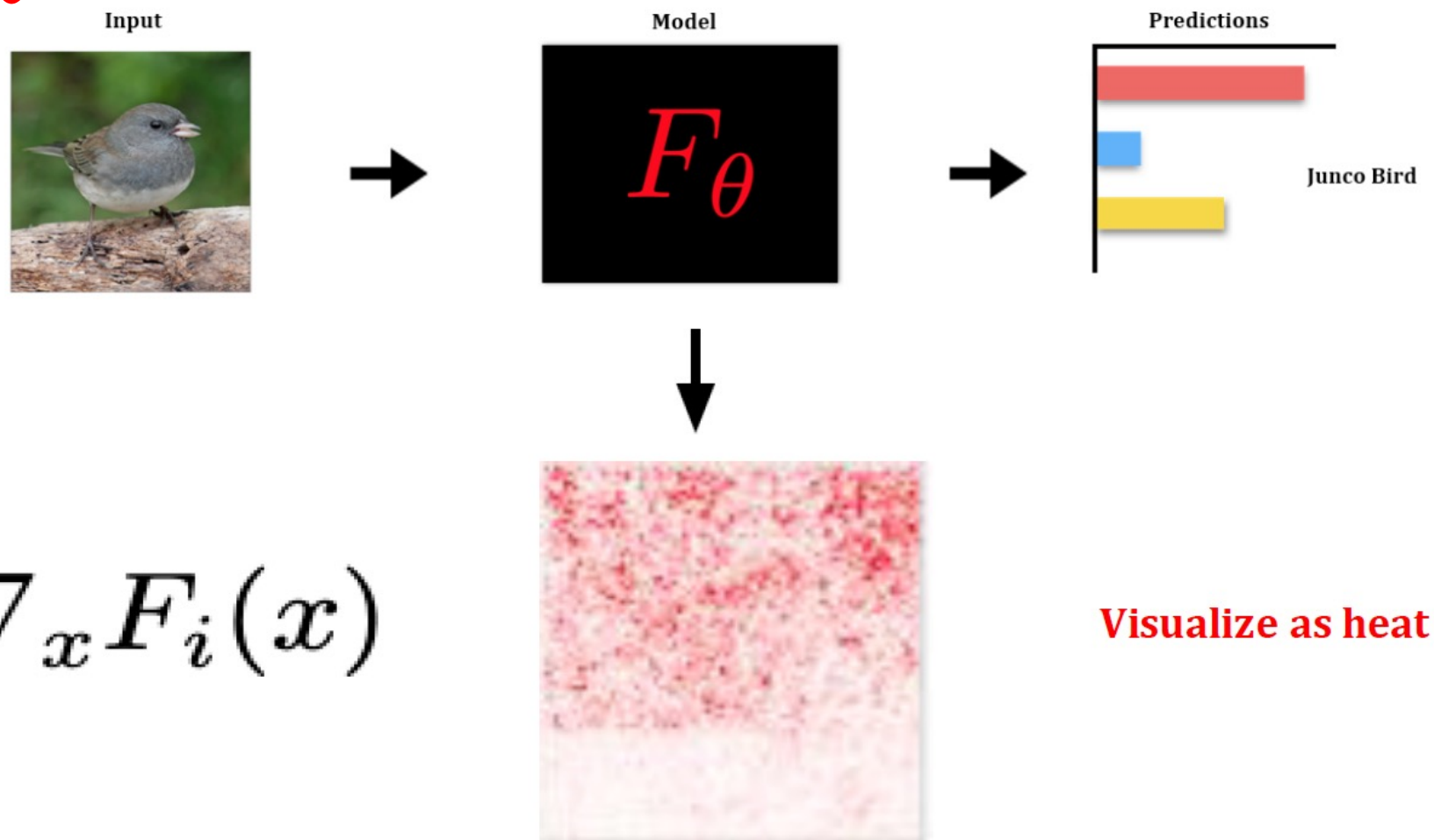
$$L(\hat{f}, g, \pi_{\mathbf{x}}) = \sum_{\mathbf{z}' \in \mathbf{Z}} [\hat{f}(h_{\mathbf{x}}(\mathbf{z}')) - g(\mathbf{z}')]^2 \pi_{\mathbf{x}}(\mathbf{z}')$$

where $\mathbf{Z}$ is the training data. This is the good old boring sum of squared errors that we usually optimize for linear models. The estimated coefficients of the model, the ϕ_j's, are the Shapley values.

Fig & Material: https://christophm.github.io/interpretable-ml-book/shapley.html

Previously on CENG7880

Input Gradient



$$\nabla_x F_i(x)$$

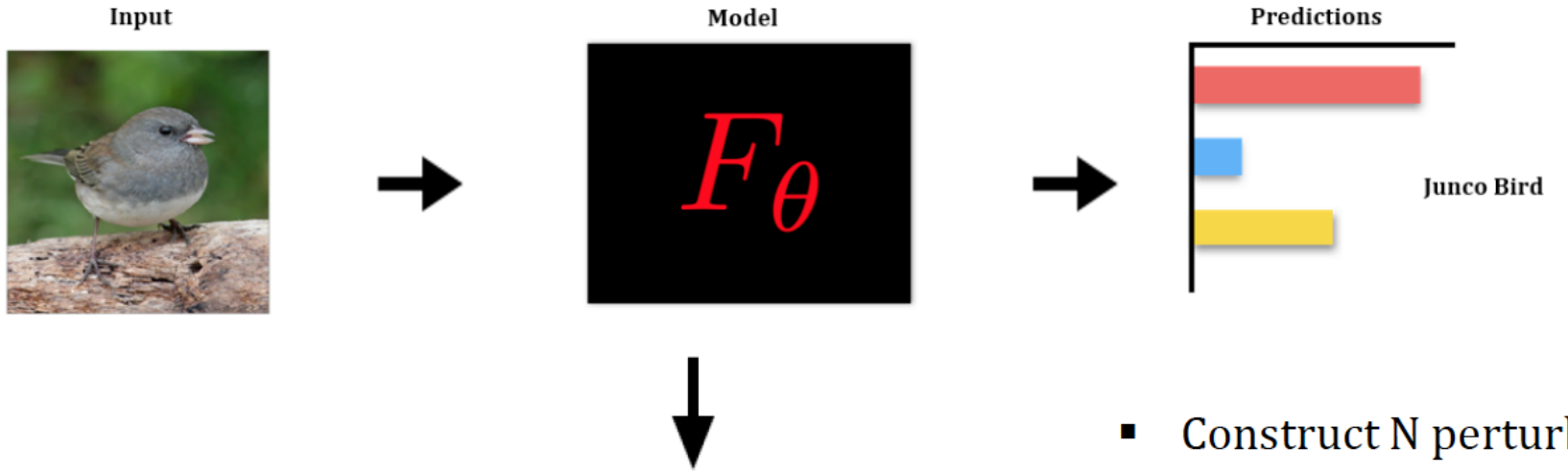
Visualize as heat map

8

SmoothGrad

Previously on CENG7880

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.



SmoothGrad

$$\frac{1}{N}\sum_{i}^{N}\nabla_{(x+\epsilon)}F_i(x+\epsilon)$$

Gaussian noise

- Construct N perturbations of input x
- By adding noise ε sampled from Gaussian distribution with standard deviation σ
- For each variant compute input gradient
- Take average

11

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Agenda

- Explainability
  - Evaluating the quality of saliency maps
  - Counterfactual explanations
  - Representation-level explanations
  - Data attribution methods
  - Concept Bottleneck Models

Administrative Notes

- Final Exam:
  - 13 January 16:30
- Paper selection finalized except for two projects
- Project milestones
  - **1. Milestone (November 23, midnight):**
    - Read & understand the paper
    - Download the datasets
    - Prepare the Readme file excluding the results & conclusion
  - **2. Milestone (December 7, midnight)**
    - The results of the first experiment
  - **3. Milestone (January 4, midnight)**
    - Final report (Readme file)
    - Repo with all code & trained models

Measuring the Quality of Feature-Attribution Weights

Saliency maps are popular for vision

Given model f: $R^n \rightarrow R$ and input x in R^n:

- α = AttributionMethod(f, x) in R^n

Simple "interpretation": if α_i is big, then x_i is important!

- Useful when the end-user may not be ML specialists

… but what does "important" mean exactly?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Feature attributions are "obvious" for simple models

Consider a linear model

$$f(x) = c_0 + c_1 x_1 + \cdots + c_n x_n$$

Clearly if the larger some c_i, the more x_i will contribute to the score

- A natural feature attribution: $\alpha_i = c_i$ (alternatively, $\alpha_i = \text{abs}(c_i)$)

… but what about for a quadratic model?

$$f(x) = c + b^\top x + x^\top A x = c + \sum_{i=1}^{n} b_i x_i + \sum_{1 \le i,j \le n} A_{ij} x_i x_j$$

It's less clear what score each feature x_i should get

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

"Fundamental dilemma" of feature attributions

Pro: feature attributions are "nice"

- Easy to understand: number big = feature important
- There's a lot of attribution methods

Cons:

- What does "important" mean?
- There's a lot of attribution methods
  - "This feature has Shapley value XXX", okay, so what?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Idea: maybe we can measure the "quality" of FAs

If a feature is "important", then it should satisfy some properties.

- … but what are these properties, and can we quantify them?

There is substantial work on developing metrics for feature attributions

- There's a lot, we'll talk about a few

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Sample Work from Our Research

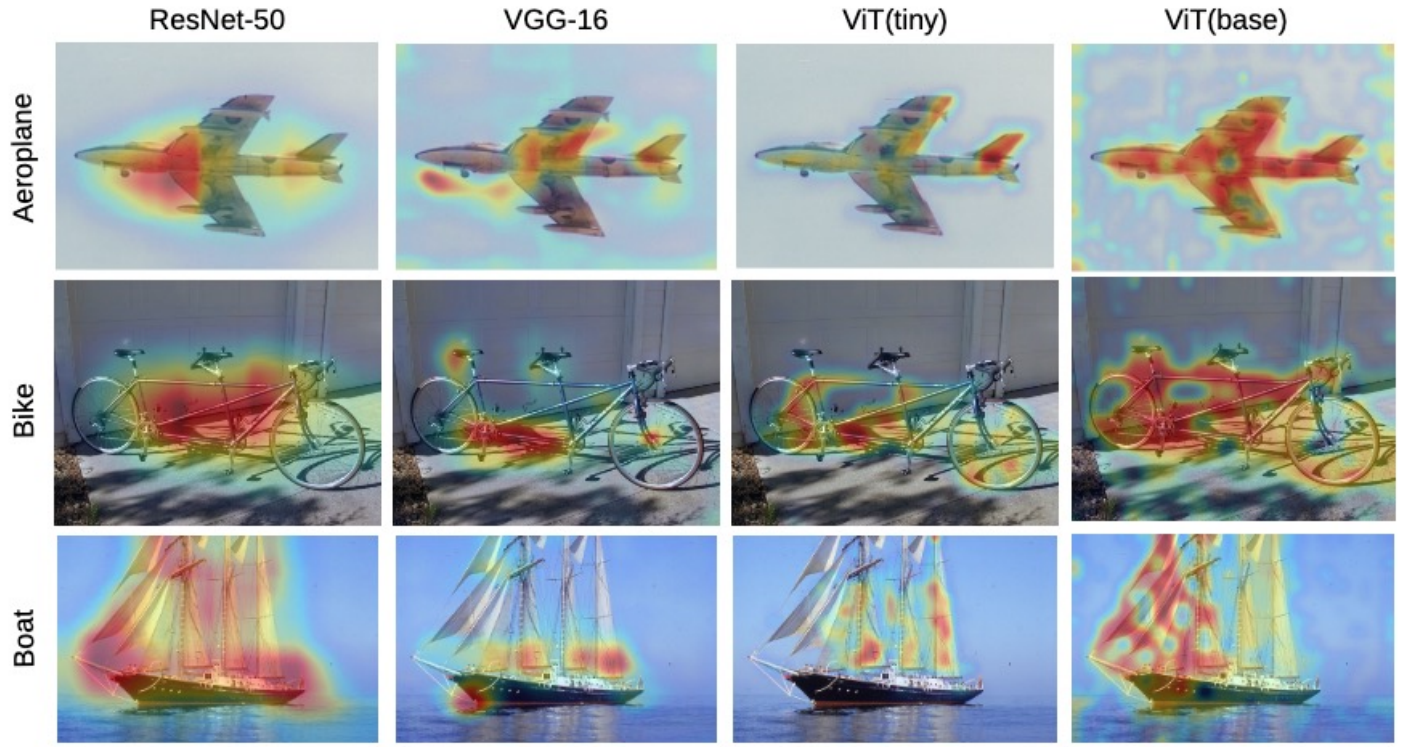


Figure 6: Exp. 1: Example heatmap visualizations. The visualization method is GradCam + SESS.

Osman Tursun, Sinan Kalkan, Simon Denman, Sridha Sridharan, Clinton Fookes, “Quantized and Verbalized Heatmap Analysis with Part-Masks and LLMs for End-User-Friendly XAI”, under review.

Sample Work from Our Research

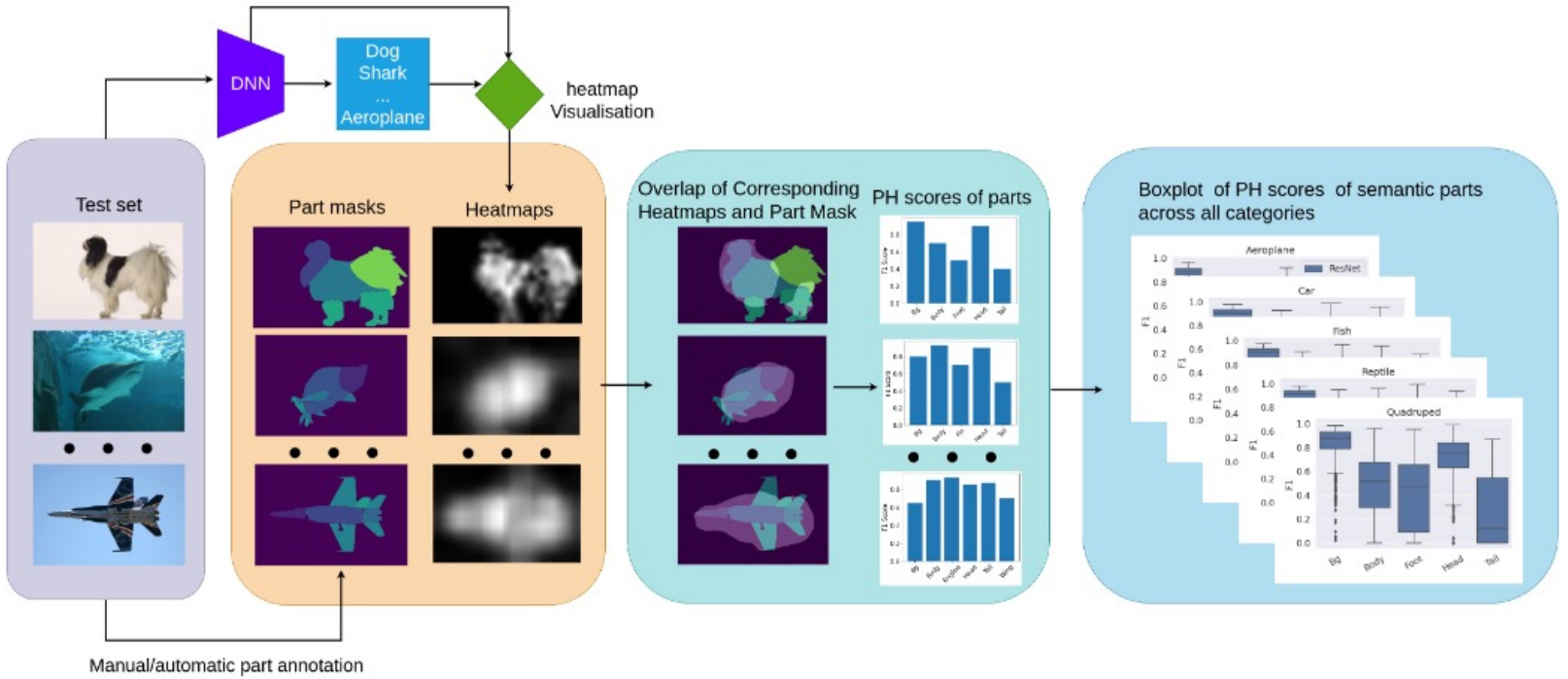
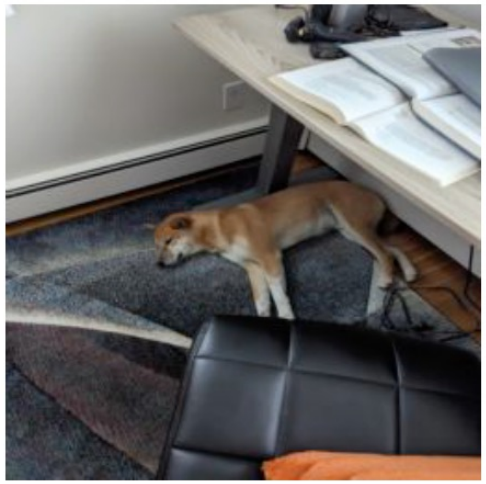


Figure 2: Overview of the Part-based Quantitative Analysis for Heatmaps (PQAH) framework. The process involves (1) extracting part masks and heatmaps from the given image dataset, (2) computing the PQAH Overlap scores for each semantic part of the main object in the images, and (3) aggregating *PH* scores across all semantic part categories to generate statistical summaries and visual representations.

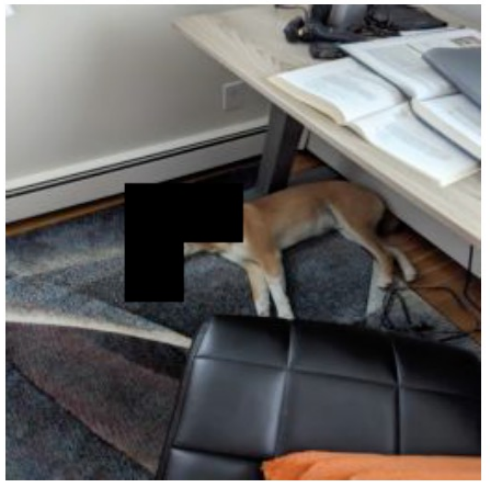
Osman Tursun, Sinan Kalkan, Simon Denman, Sridha Sridharan, Clinton Fookes, "Quantized and Verbalized Heatmap Analysis with Part-Masks and LLMs for End-User-Friendly XAI", under review.

Subtractive metrics

"If some feature is important, then removing it should decrease the score"



"Dog" (97%)



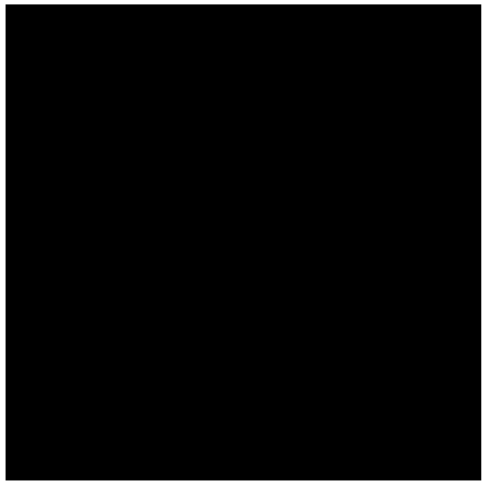
"Dog" (50%)

*I made up these numbers

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Additive metrics

"If a feature is important, then inserting it should increase the score"



"Dog" (<1%)

"Dog" (40%)

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Example of other metrics

Perturbation:

- How sensitive is your metric to perturbations?

Compactness:

- Is your explanation too "big"? (e.g., for feature selection)

Connectedness:

- Are two candidate explanations "connected" in some sense?

More here: https://arxiv.org/abs/2201.08164

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

What mathematical properties should we expect?

Given a model, an input, an explanation method, and some metric …

… what formal (i.e., mathematical) properties should these things satisfy?

- e.g., "does the top-k% of features from this method guarantee a score decrease of q% wrt some metric, model class, and input family?"

In general? Hard to prove such statements

- Neural networks are magic

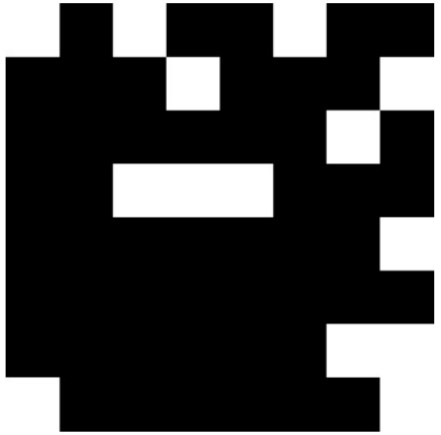
Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

What can we do from here?

Our work: under some conditions, one CAN guarantee formal properties

Special case: binary-valued feature attributions (i.e., feature selection)

Input x in R^n

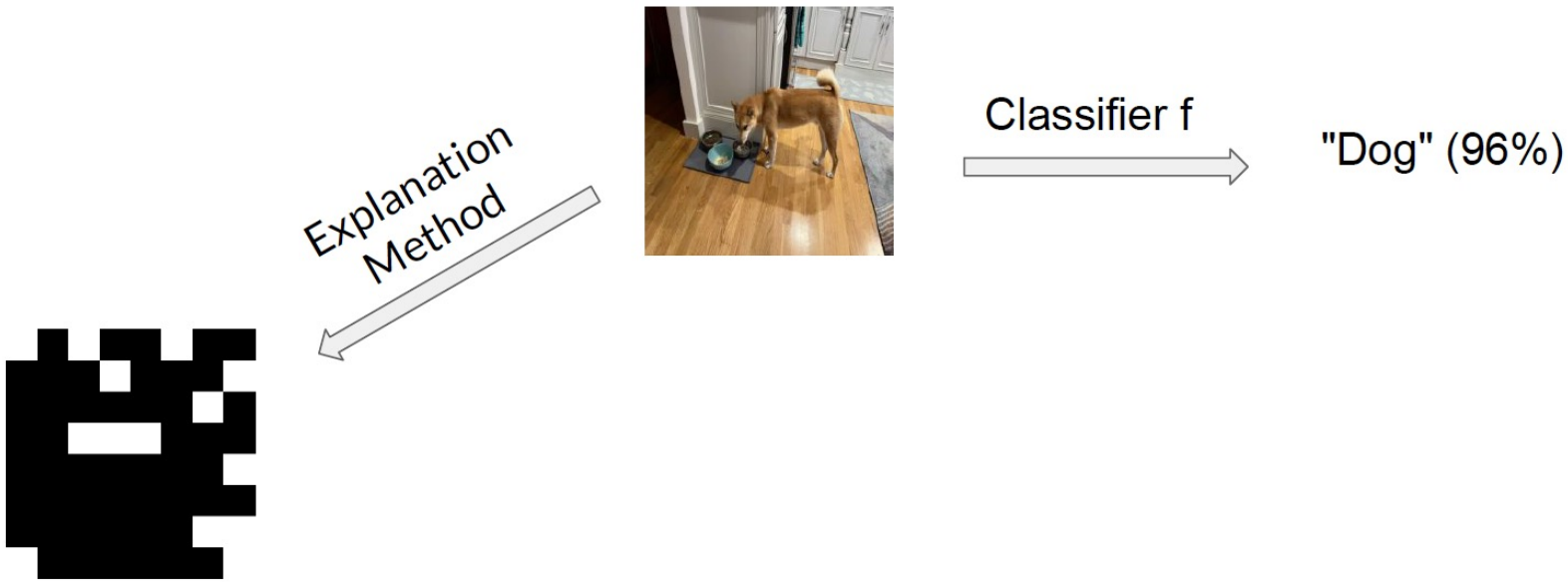


Attr α in $\{0,1\}^n$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

I have an attribution, but how do I "evaluate" it?

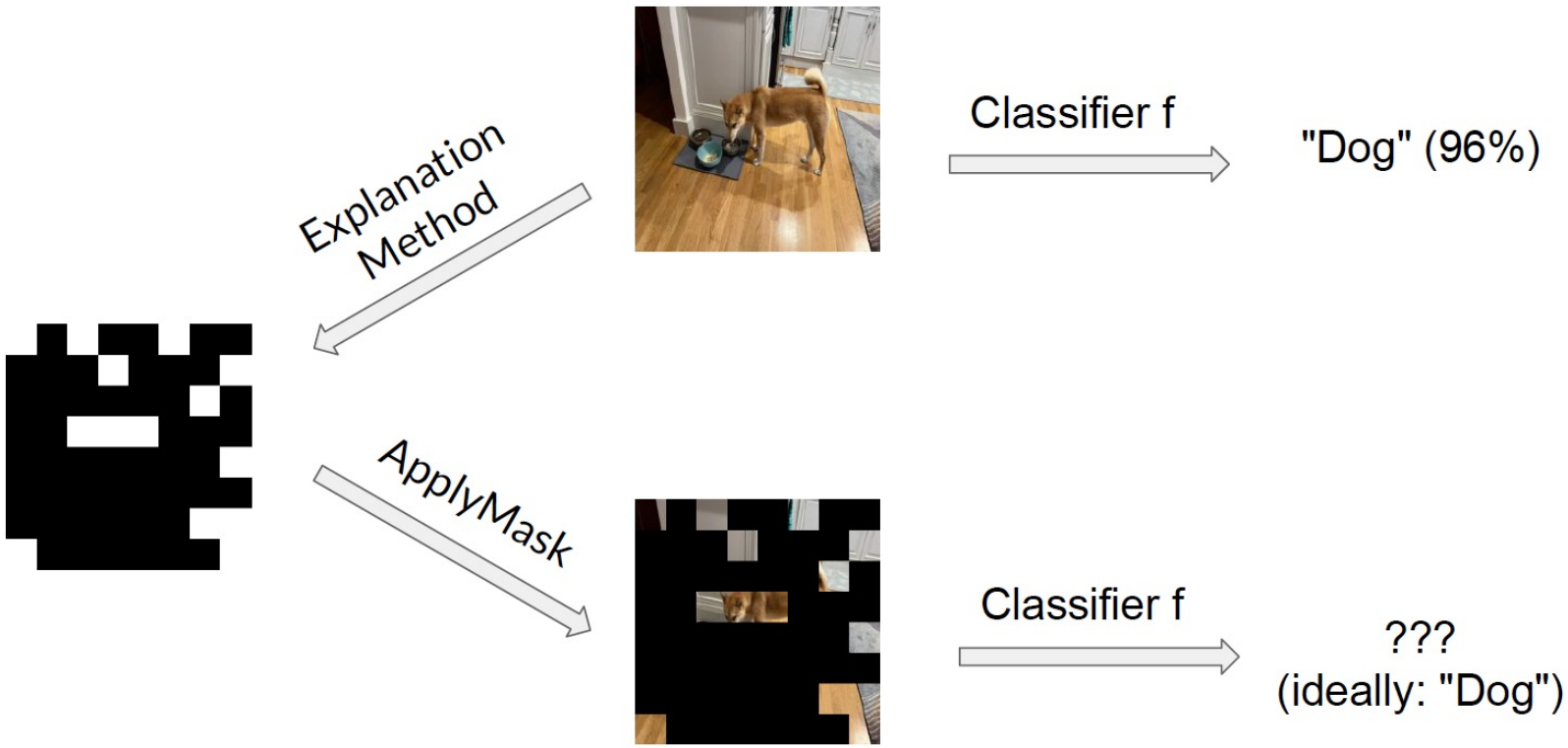
In vision: we can use the original model



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

I have an attribution, but how do I "evaluate" it?

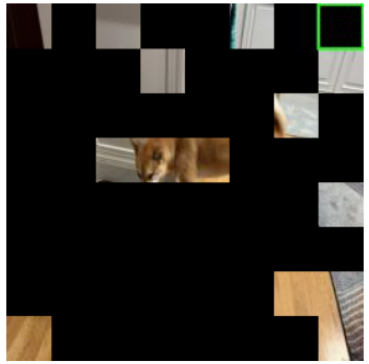
In vision: we can use the original model



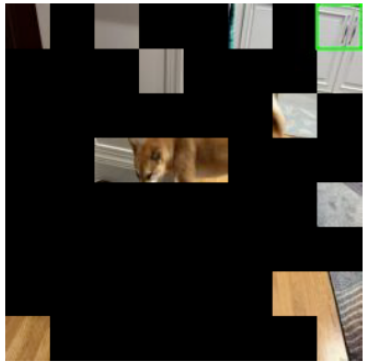
Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

What do we NOT want to happen?

"Dog"



"Dog"



"Cat"

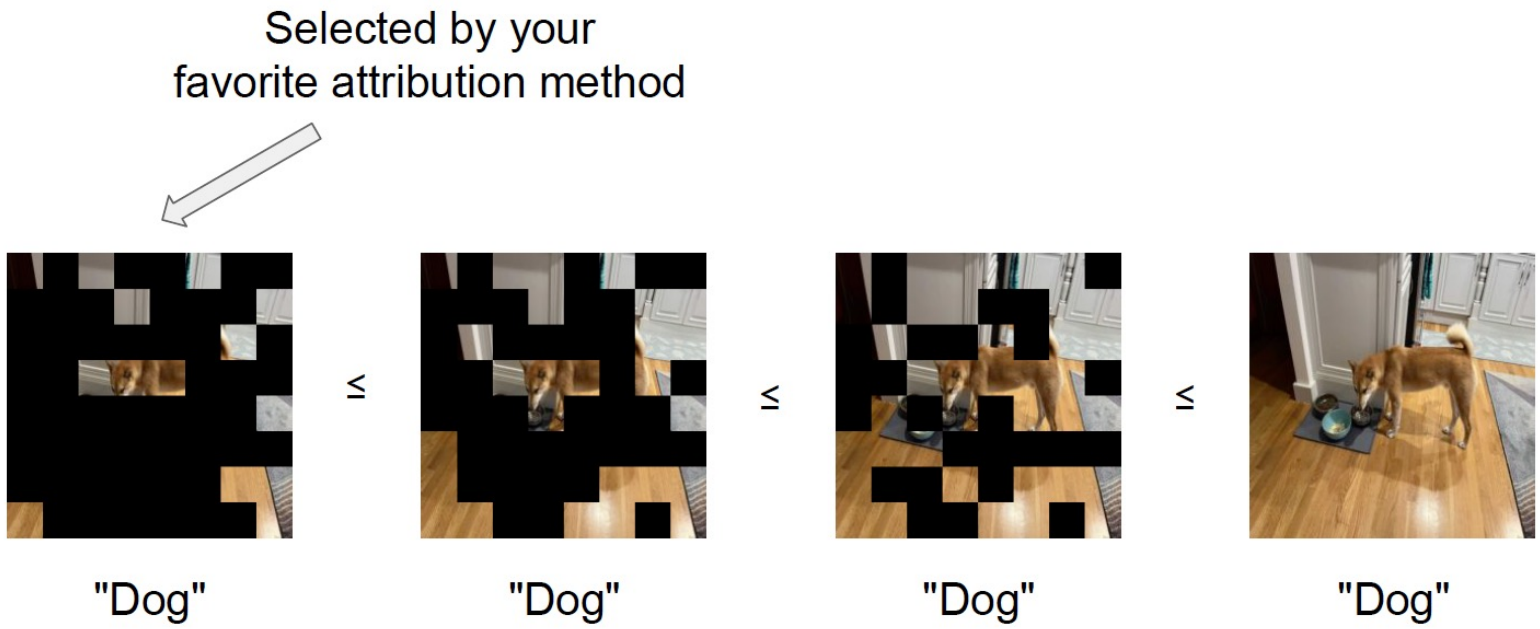
This is usually **NOT** desirable

Intuition: the original feature selection you gave me is not "convincing"

*I made up this example, but it can happen. Trust me, bro!

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Stability as a "desirable" property



Stability: any superset of features induces the same prediction

$f(x \circ \alpha) \cong f(x \circ \alpha')$ for all $\alpha \leq \alpha'$, where α = BinaryAttribution(f,x)

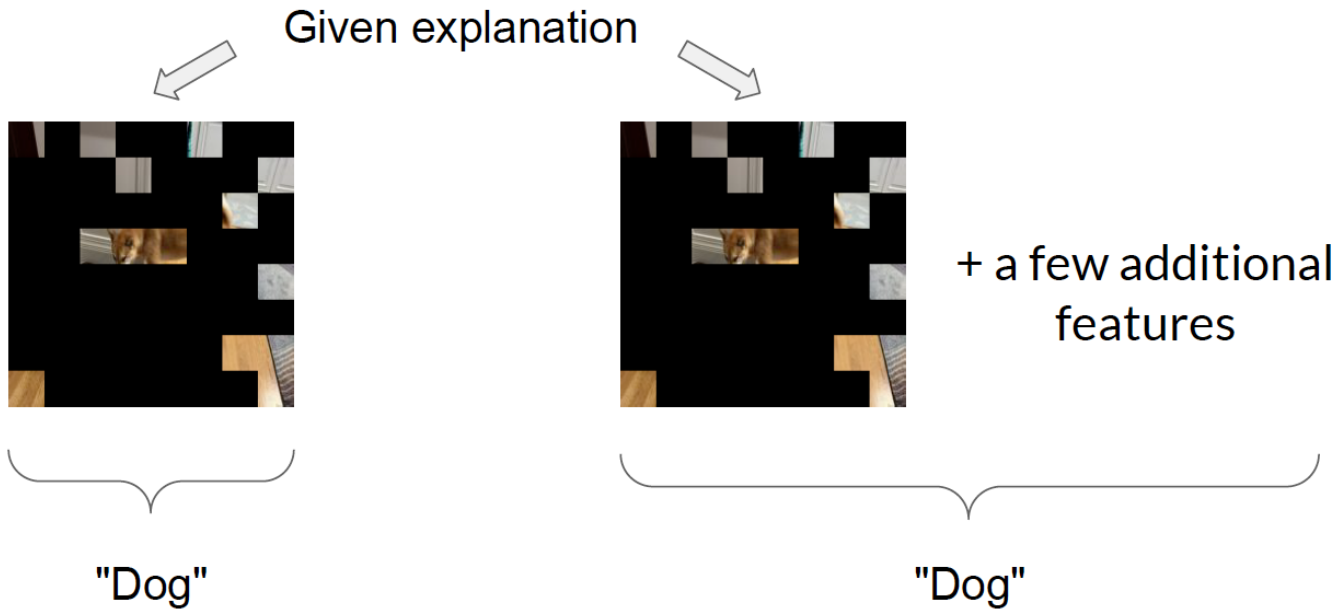
Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

How can we achieve/guarantee stability!

You probably can't! (For Real Models^{TM})

- There's $O(2^n)$ different $\alpha' \geq \alpha$ binary vectors to check

But we can maybe go for local approximations (Incremental Stability)

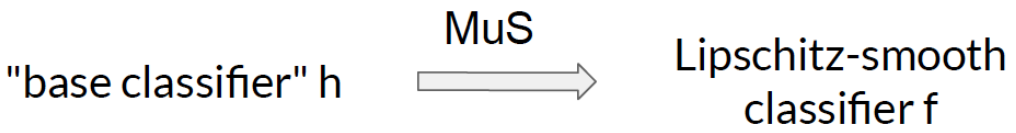


Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

The plan

1. Incremental stability via Lipschitz smoothness

2. Achieve incremental stability with multiplicative smoothing (MUS)



3. We can check if f is incrementally stable at some x in O(1) time.

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Step 1: incremental stability

f() ≅ f( +Δ) for all small Δ

Sufficient condition: Lipschitz wrt masking of features

- L1 norm on binary vectors = number of differences

f() - f() ≤ λ||  -  $||_1$ for all  , 

Definition (Lipschitz wrt Feature Maskings). The function f: $R^n \rightarrow [0,1]$ is λ-Lipschitz wrt the masking of features at x in R^n if:

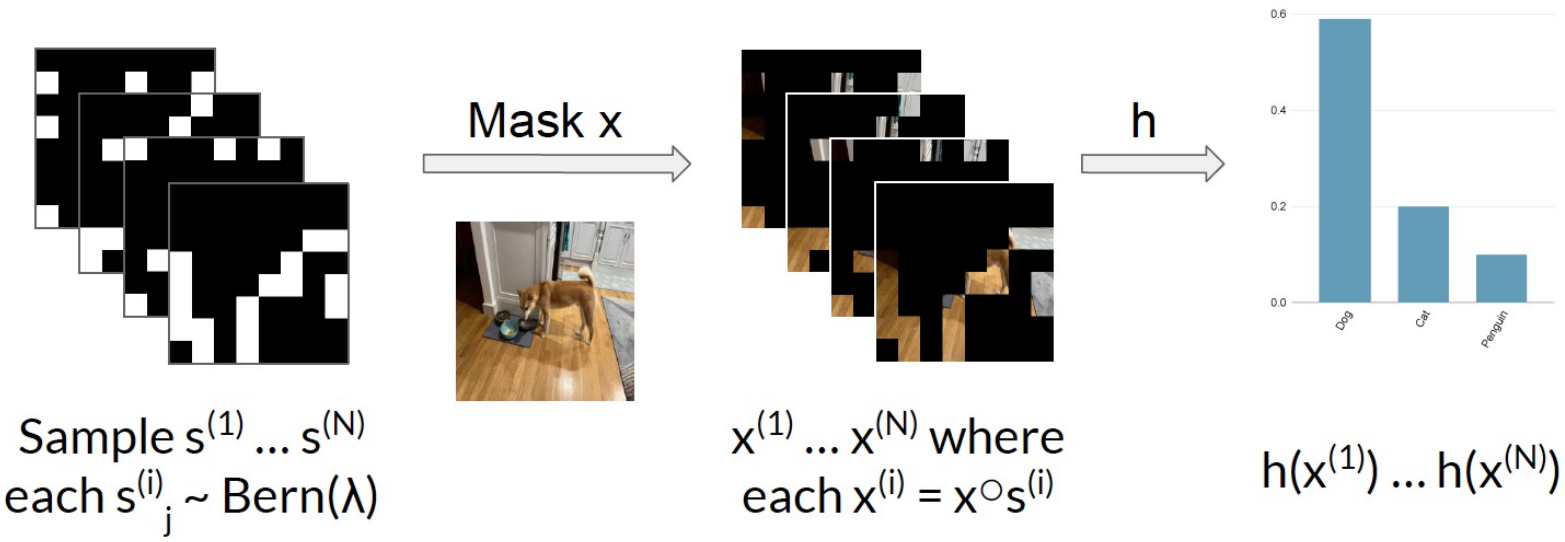
$$f(x \circ \alpha) - f(x \circ \alpha') \leq \lambda ||\alpha - \alpha'||_1 \quad \text{for all } \alpha, \alpha' \text{ in } \{0,1\}^n$$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Step 2: Multiplicative Smoothing (MuS)

"base classifier" h $\xrightarrow{\text{MuS}}$ λ-Lipschitz-smooth f

$$f(x) = \text{MuS}(h, x) = \text{avg}(h(x^{(1)}), \ldots, h(x^{(N)}))$$



Sample $s^{(1)} \ldots s^{(N)}$
each $s^{(i)}_j \sim \text{Bern}(\lambda)$

$x^{(1)} \ldots x^{(N)}$ where
each $x^{(i)} = x \circ s^{(i)}$

$h(x^{(1)}) \ldots h(x^{(N)})$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Step 2: The Math

Recall f(x) = MuS(h, x) and let

$$g(x, \alpha) = \text{MuS}(h, x \circ \alpha) = \text{avg}(h(x \circ \alpha \circ s^{(1)}), \ldots, h(x \circ \alpha \circ s^{(N)}))$$

Theorem (MuS). Let D be any distribution on $\{0,1\}^n$ where each coordinate of $s \sim D$ is marginally distributed as $s_i \sim \text{Bern}(\lambda)$ and let

$$g(x, \alpha) = E_{s \sim D}\, h(x \circ \alpha \circ s), \qquad \text{for any } h: R^n \to [0,1],$$

then $g(x, \alpha)$ is λ-Lipschitz in α wrt the L^1 norm for all x.

Note: Lipschitz smoothness is a property of functions

D need NOT be coordinate-wise independent

- We just requires that each sample's coordinate marginally ~ Bern(λ)
- Allows for a deterministic evaluation with $N \ll 2^n$ samples
  - Recall that $\text{Bern}^n(\lambda)$ has 2^n unique values

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Step 3: provable incremental stability

Suppose h: $R^n \rightarrow [0,1]^m$ is a classifier

Let f(x) = MuS(h, x) with parameter λ

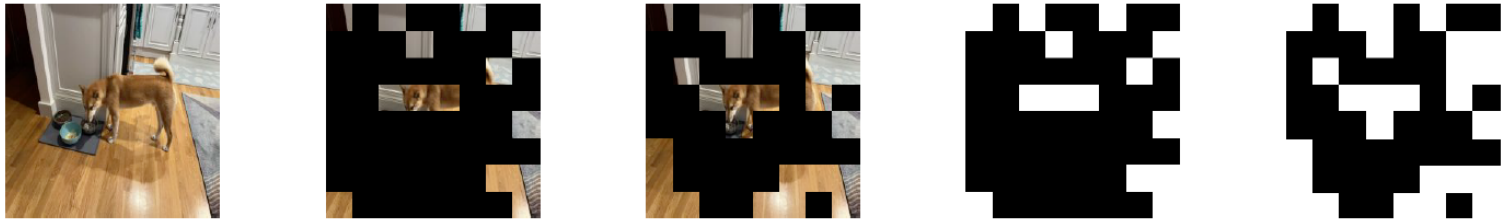
Probability of the Bernouilli distribution (in the previous slide)

Let α = BinaryAttribution(f, x), such that $f(x) \cong f(x \circ \alpha)$

Theorem (MuS). Suppose that

$$\text{Class1Prob}(f(x \circ \alpha)) - \text{Class2Prob}(f(x \circ \alpha)) \geq 2\lambda r,$$

then for any $\alpha' \geq \alpha$ with $||\alpha' - \alpha||_1 \leq r$, we have $f(x \circ \alpha') \cong f(x \circ \alpha)$.



x $\quad$ $x \circ \alpha$ $\quad$ $x \circ \alpha'$ $\quad$ α $\quad$ α'

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Basic summary of MuS

Step 1: stability is hard, so we go for incremental stability

- Key idea: Lipschitz constants

Step 2: "randomized" smoothing

- $f(x) = MuS(h, x) = avg(h(x \circ s^{(1)}), \ldots, h(x \circ s^{(N)}))$
- $f(x \circ \alpha) = MuS(h, x \circ \alpha) = avg(h(x \circ \alpha \circ s^{(1)}), \ldots, h(x \circ \alpha \circ s^{(N)}))$

Step 3: Lipschitz constants → stability guarantees

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Counterfactuals

Background: Counterfactual Reasoning

The Ladder of Causality

"Actual" Causality

"Causality-in-mean"

Statistics

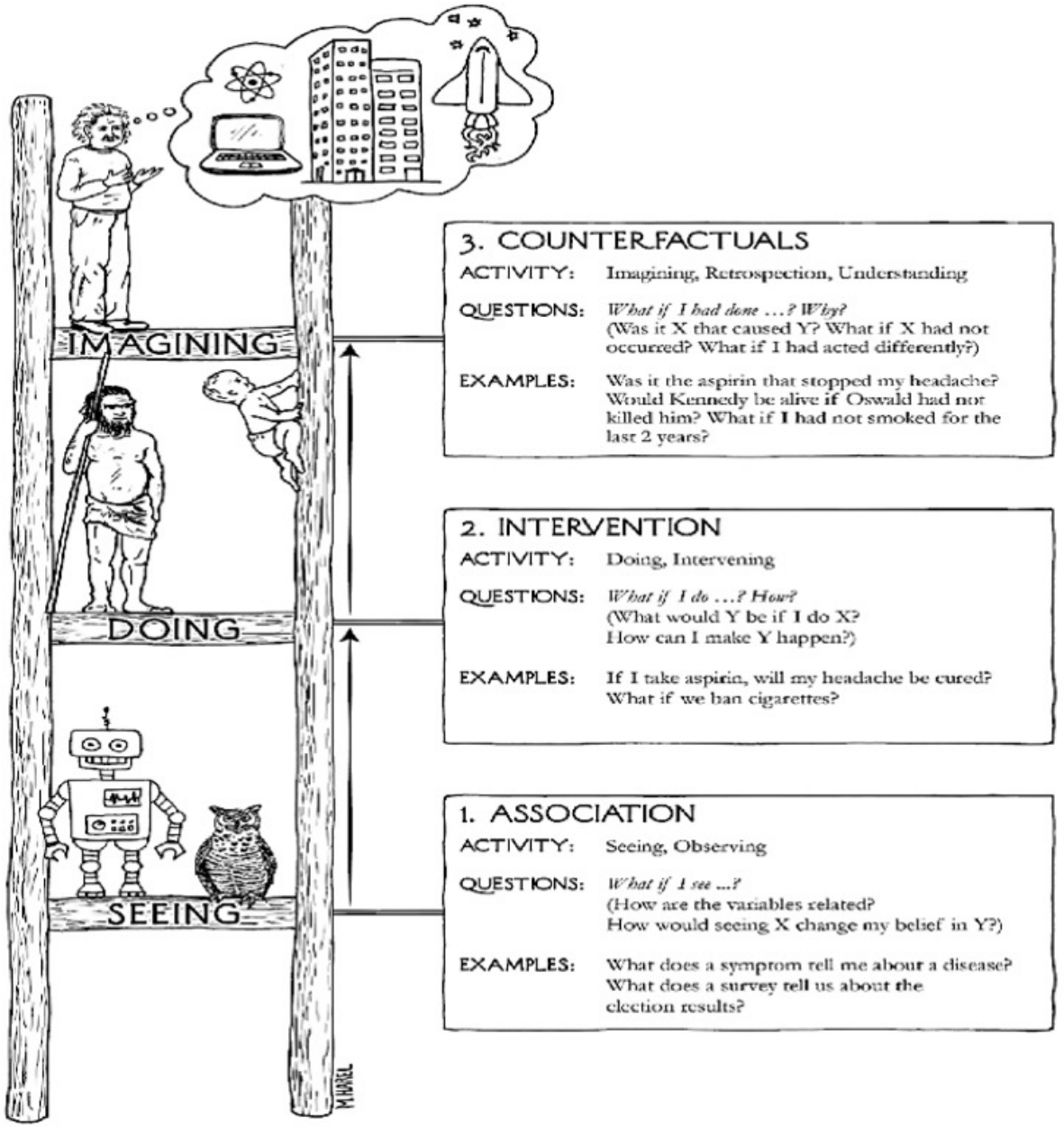


Figure: Judea Pearl.

Counterfactual

- As ML models are increasingly deployed to make high-stakes decisions (e.g., loan applications), it becomes important to provide recourse to affected individuals.

> ***Counterfactual Explanations***
> *What features need to be changed and by how much to flip a model's prediction ?*
> *(i.e., to reverse an unfavorable outcome).*

- Key publications:
  - Counterfactual visual explanations; Goyal et al.; ICML 2019
  - Counterfactual explanations without opening the black box; Wachter et al.; 2018

3

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

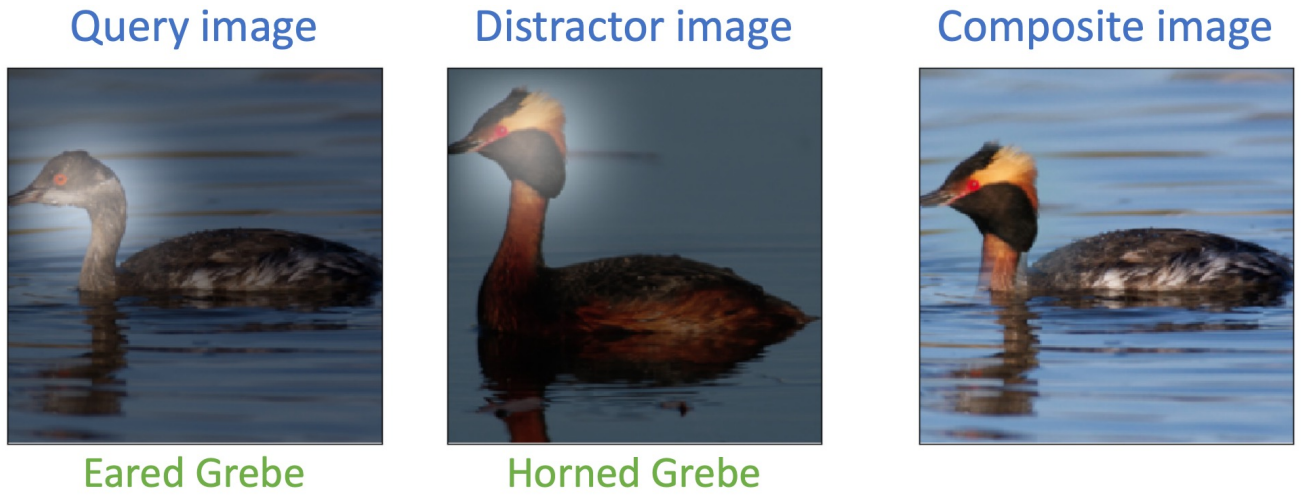
Counterfactual Explanations

- When trying to understand/explain model behaviors, we often want to answer questions

  "For situation X, why was the outcome Y and not Z?"

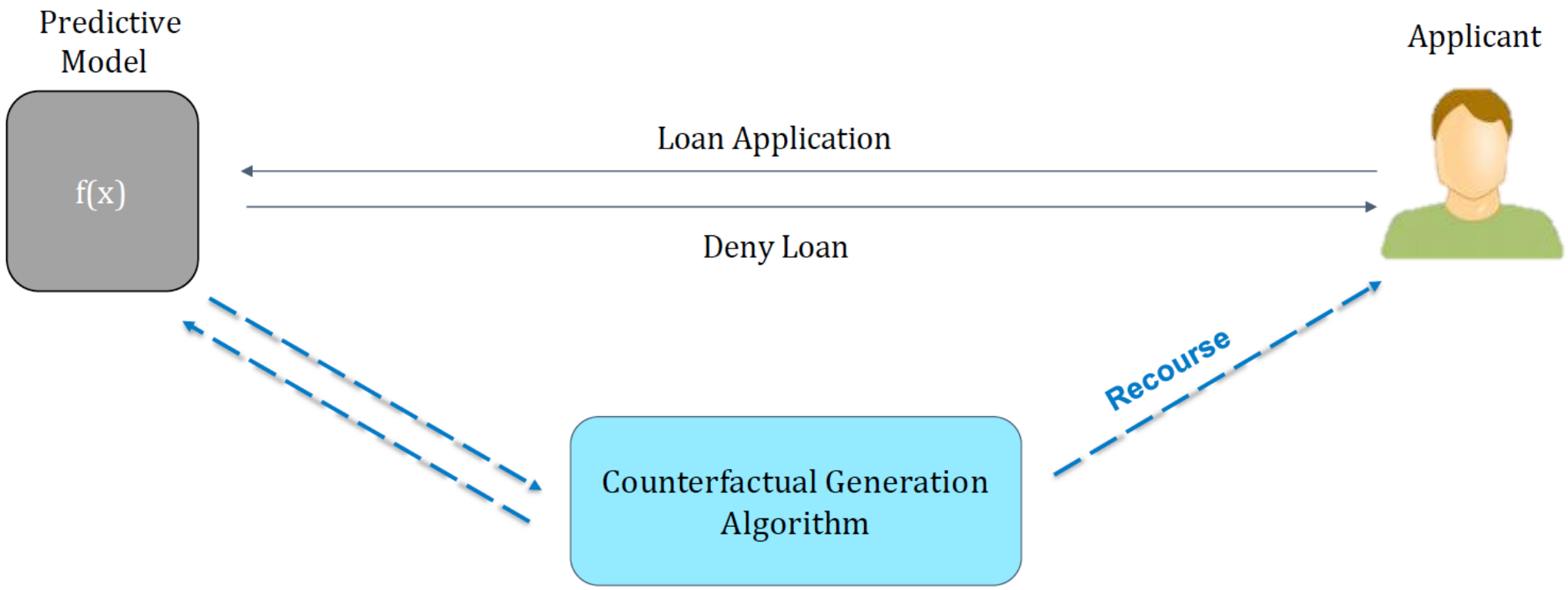
- One way to answer this is to construct a counterfactual sample that includes/contains the potential cause:

  "If X was X*, then the outcome would have been Z rather than Y."



Goyal et al., Counterfactual Visual Explanations, 2019.

Counterfactual Explanation

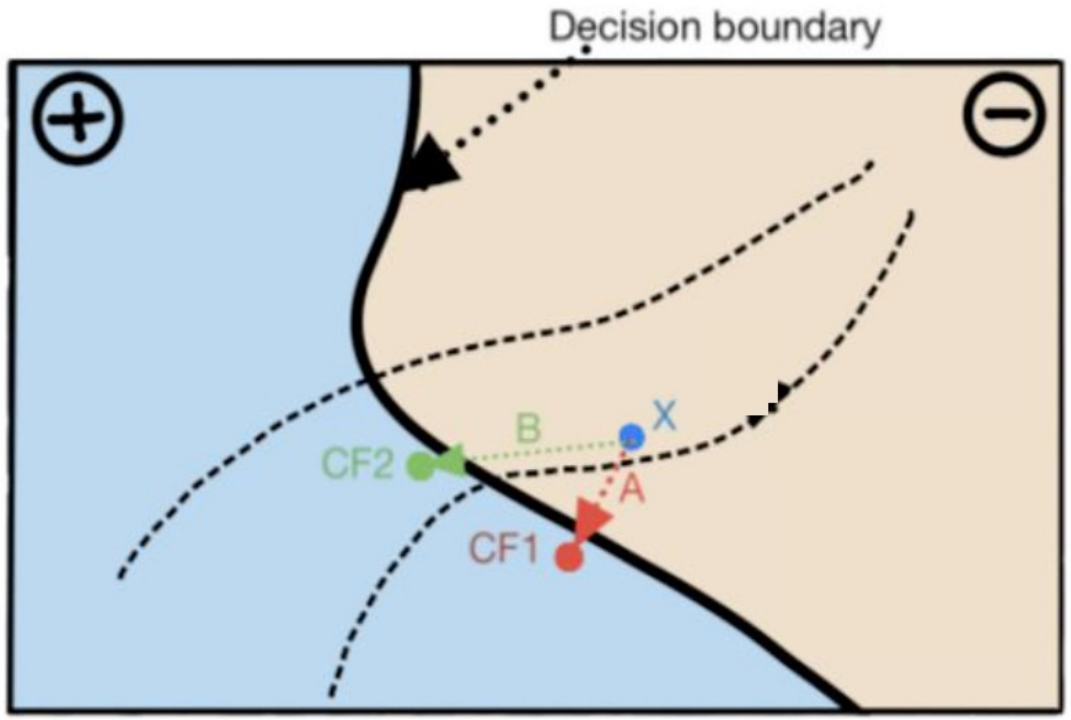


Recourse: Increase your salary by 5K & pay your credit card bills on time for next 3 months

4

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Generating Counterfactual

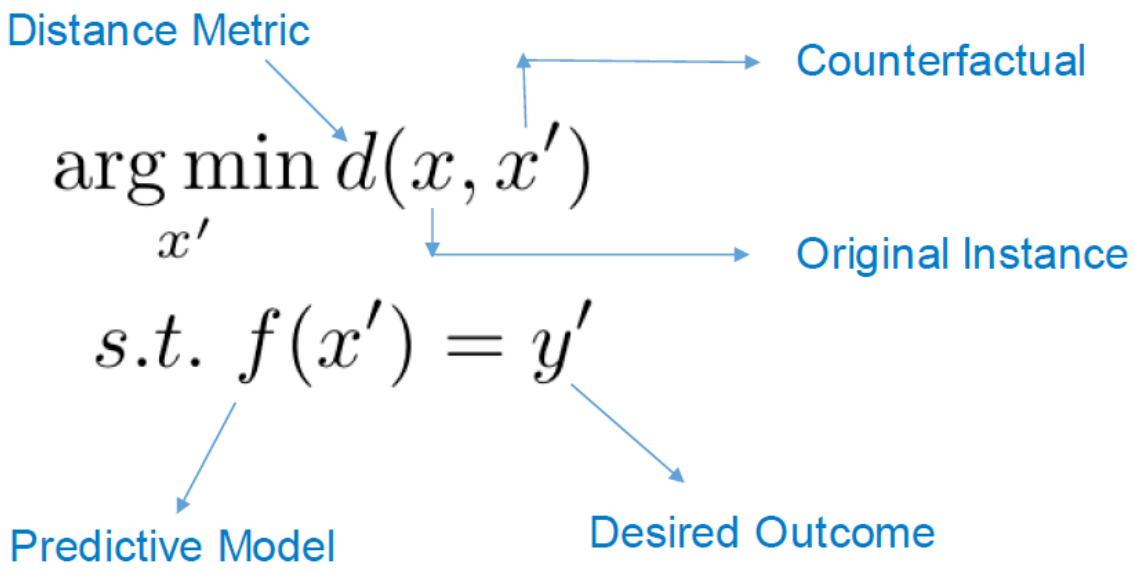


Proposed solutions differ on:

- How to choose among candidate counterfactuals?
- How much access is needed to the underlying predictive model?

5

Minimum Distance Counterfactuals



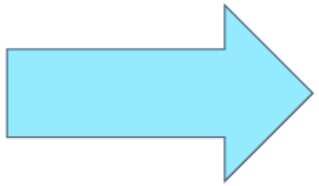
Choice of distance metric dictates what kinds of counterfactuals are chosen.

Wachter et al. (2018) use normalized Manhattan distance.

6

Minimum Distance Counterfactual Computation

$$\arg\min_{x'} d(x, x') \quad s.t.\ f(x') = y'$$



$$\arg\min_{x'} \lambda\ (f(x') - y')^2 + d(x, x')$$

Wachter et. al. solve a differentiable, unconstrained version of the objective using ADAM optimization algorithm with random restarts.

This method ***requires access to gradients*** of the underlying predictive model.

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Feasibility Challenge

Person 1: If your LSAT was 34.0, you would have an average predicted score (0).

Person 2: If your LSAT was 32.4, you would have an average predicted score (0).

Person 3: If your LSAT was 33.5, and you were 'white', you would have an average predicted score (0).

Person 4: If your LSAT was 35.8, and you were 'white', you would have an average predicted score (0).

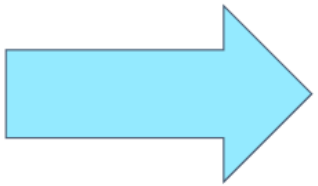
Person 5: If your LSAT was 34.9, you would have an average predicted score (0).

Not feasible to act upon these features!

8

Feasible Least Cost Counterfactuals

$$\arg\min_{x'} d(x, x') \quad s.t.\ f(x') = y'$$



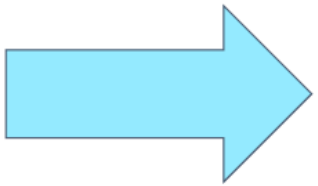
$$\arg\min_{x' \in \mathcal{A}} cost(x, x') \quad s.t.\ f(x') = y'$$

- $\mathcal{A}$ is the set of feasible counterfactuals (input by end user)
  E.g., changes to race, gender are not feasible
- *Cost* is modeled as total log-percentile shift
  Changes become harder when starting off from a higher percentile value

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Feasible Least Cost Counterfactuals

$$\arg\min_{x'} d(x, x') \quad s.t.\ f(x') = y'$$



$$\arg\min_{x' \in \mathcal{A}} cost(x, x') \quad s.t.\ f(x') = y'$$

How to solve such an optimization problem ?

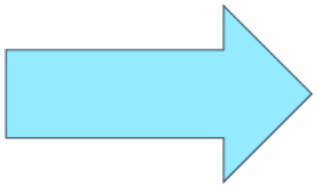
When model f is linear, use ILP (integer linear programming)

10

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Feasible Least Cost Counterfactuals

$$\arg\min_{x'} d(x, x') \quad s.t.\ f(x') = y'$$



$$\arg\min_{x' \in \mathcal{A}} cost(x, x') \quad s.t.\ f(x') = y'$$

How to handle non-linear classifiers ?

Generate a local linear model approximation (e.g. using LIME) around input x

11

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

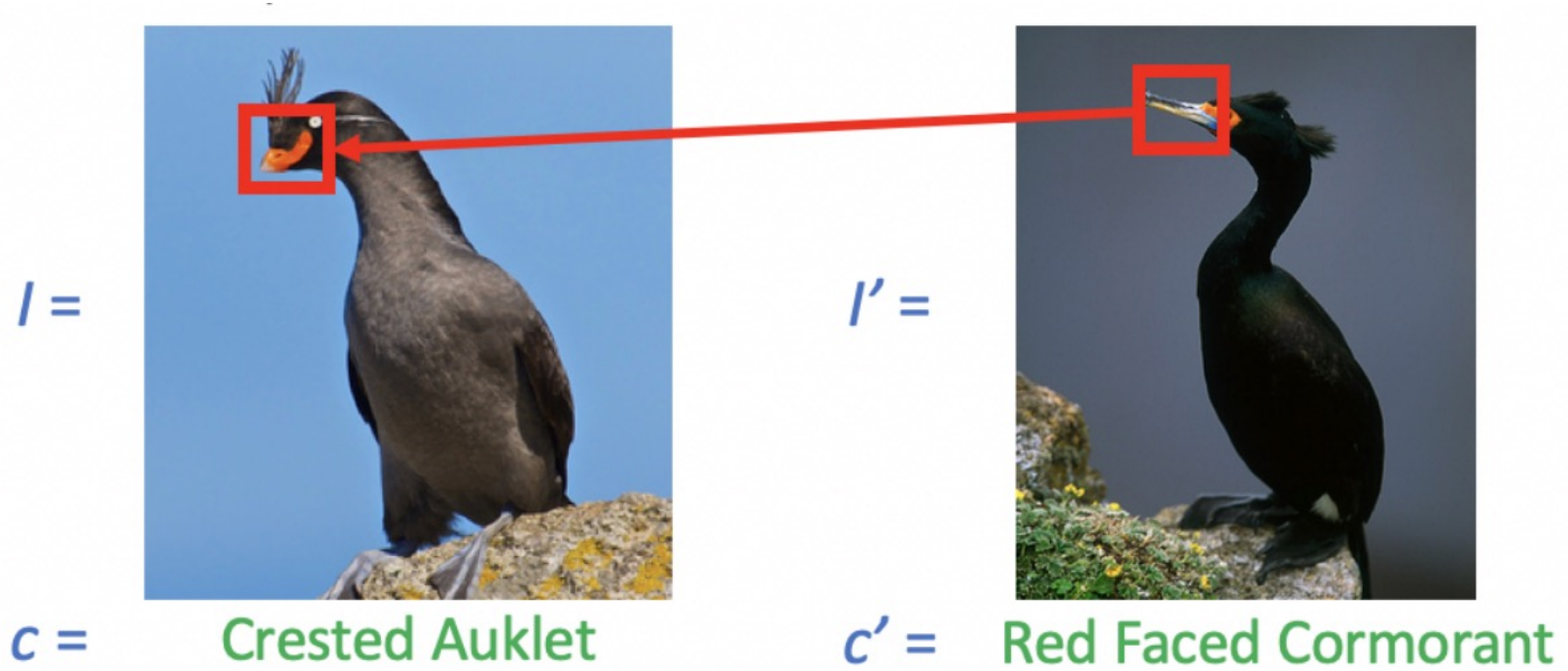


Figure 1. Our approach generates counterfactual visual explanations for a query image I (left) – explaining why the example image was classified as class c (*Crested Auklet*) rather than class c' (*Red Faced Cormorant*) by finding a region in a distractor image I' (right) and a region in the query I (highlighted in red boxes) such that if the highlighted region in the left image looked like the highlighted region in the right image, the resulting image I^* would be classified more confidently as c'.

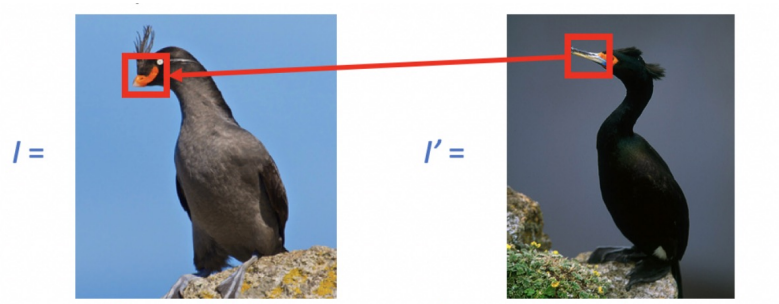
Goyal et al., Counterfactual Visual Explanations, 2019.

Visual Counterfactual Generation Problem

- Input images: source image I and target image I' (called "distractor" image)
  - Each image has dimension (h x w) x d [height h, width w, d channels]

- Model f: deep convolutional neural network mapping images to log-probability outputs in Y
  - Class predicted for source image I = c
  - Class predicted for target image I' = c'

- Goal: Find image I* obtained from source image I by replacing parts of it with those from distractor image I' such that prediction of f on I* is target class c'



13

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Visual Counterfactual Generation problem

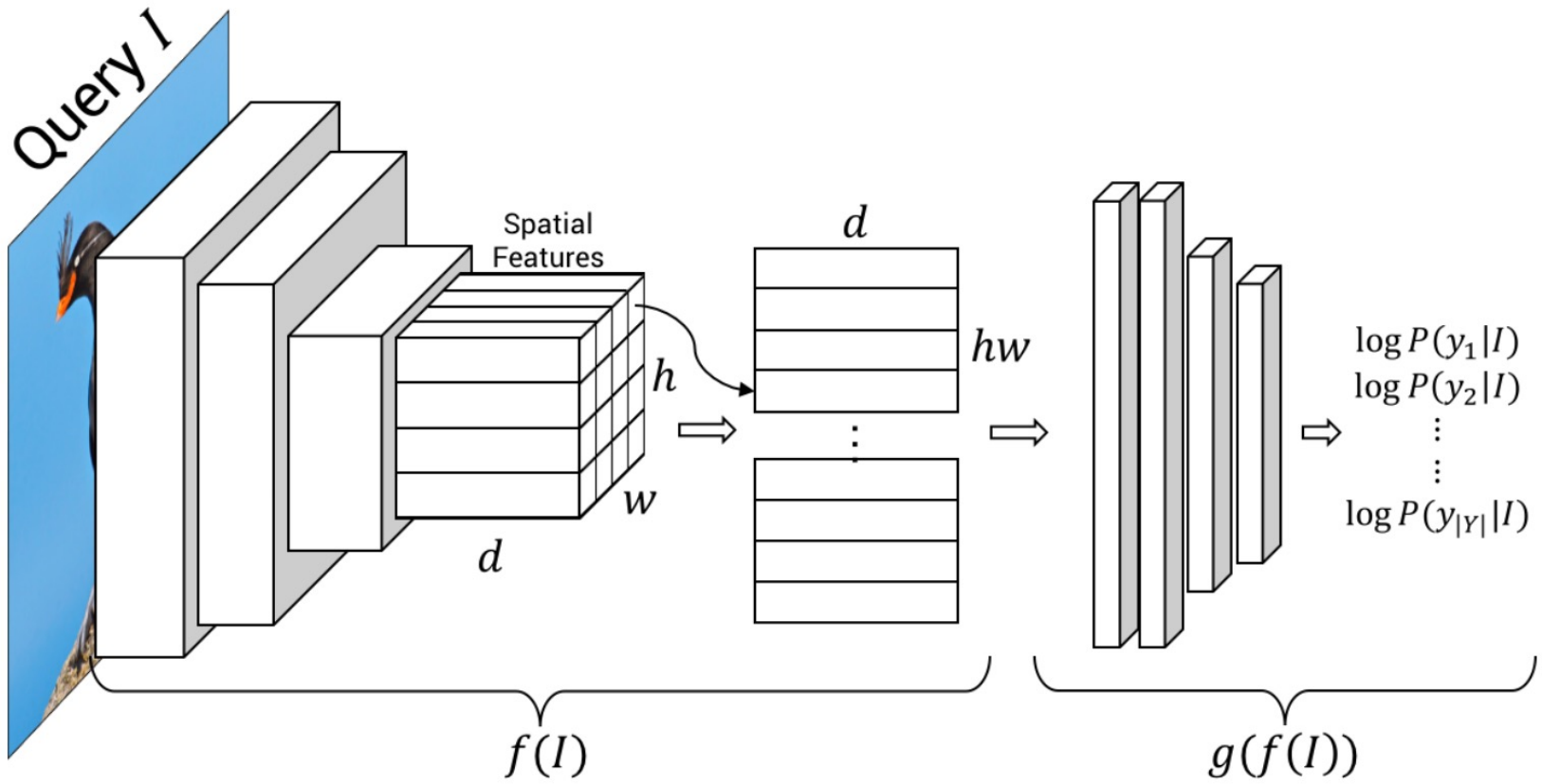


Figure 3. We decompose a CNN as a spatial feature extractor $f(I)$ and a decision network $g(f(I))$ as shown above.

14

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Minimum-edit Counterfactual

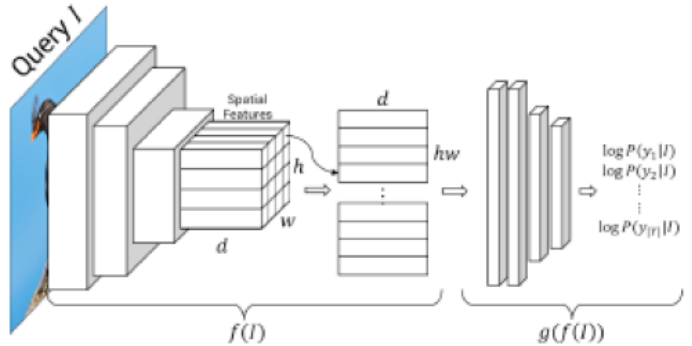


Figure 3. We decompose a CNN as a spatial feature extractor $f(I)$ and a decision network $g(f(I))$ as shown above.

- f(I') captures distractor image features
- Find [hw x hw] dimensional permutation matrix P to obtain rearranged distractor features P.f(I')
- Select a subset of these features and substitute these in f(I)
  - Corresponds to finding a gating vector a (binary vector of size hw)

$$f(I^*) = (\mathbb{1} - \mathbf{a}) \circ f(I) + \mathbf{a} \circ P f(I')$$

Transformation from I to I' in feature space.

  - We want to minimize the number of 1's (= number of edits to image I) in the gating vector a

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Minimum-edit Counterfactual

$$\underset{P,\mathbf{a}}{\text{minimize}} \quad ||a||_1$$

$$\text{s.t.} \quad c' = \text{argmax} \; g((\mathbb{1} - \mathbf{a}) \circ f(I) + \mathbf{a} \circ Pf(I'))$$

$$a_i \in \{0, 1\} \;\; \forall i \;\; \text{and} \;\; P \in \mathcal{P}$$

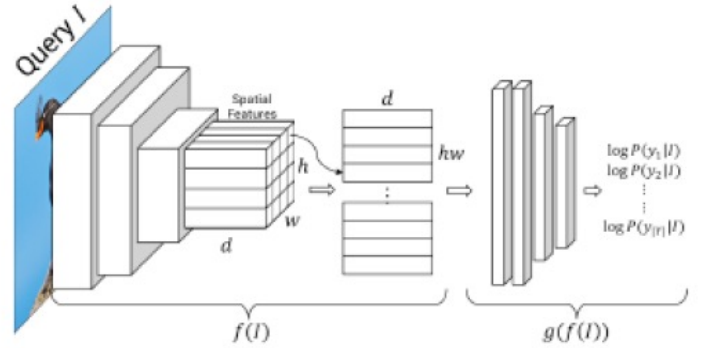


Figure 3. We decompose a CNN as a spatial feature extractor $f(I)$ and a decision network $g(f(I))$ as shown above.

reshuffle features

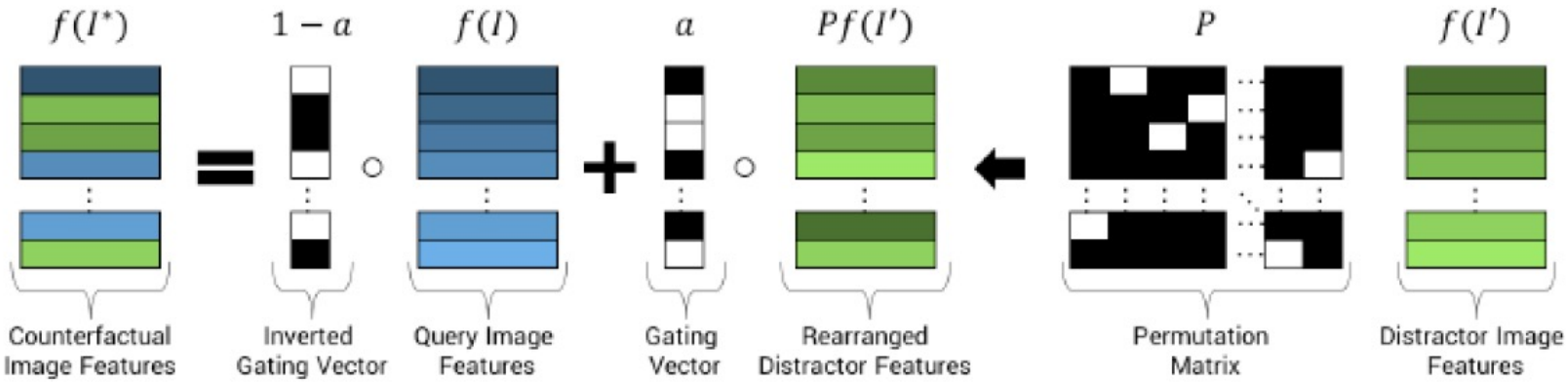


Figure 2. To parameterize our counterfactual explanations, we define a transformation that replaces regions in the query image I with those from a distractor I'. Distractor image features $f(I')$ are first rearranged with a permutation matrix P and then selectively replace entries in $f(I)$ according to a binary gating vector a. This allows arbitrary spatial cells in $f(I')$ to replace arbitrary cells in $f(I)$.

Greedy Search

minimum edit from I to I' in feature space.

$$\underset{P,\mathbf{a}}{\text{minimize}} \quad ||a||_1$$

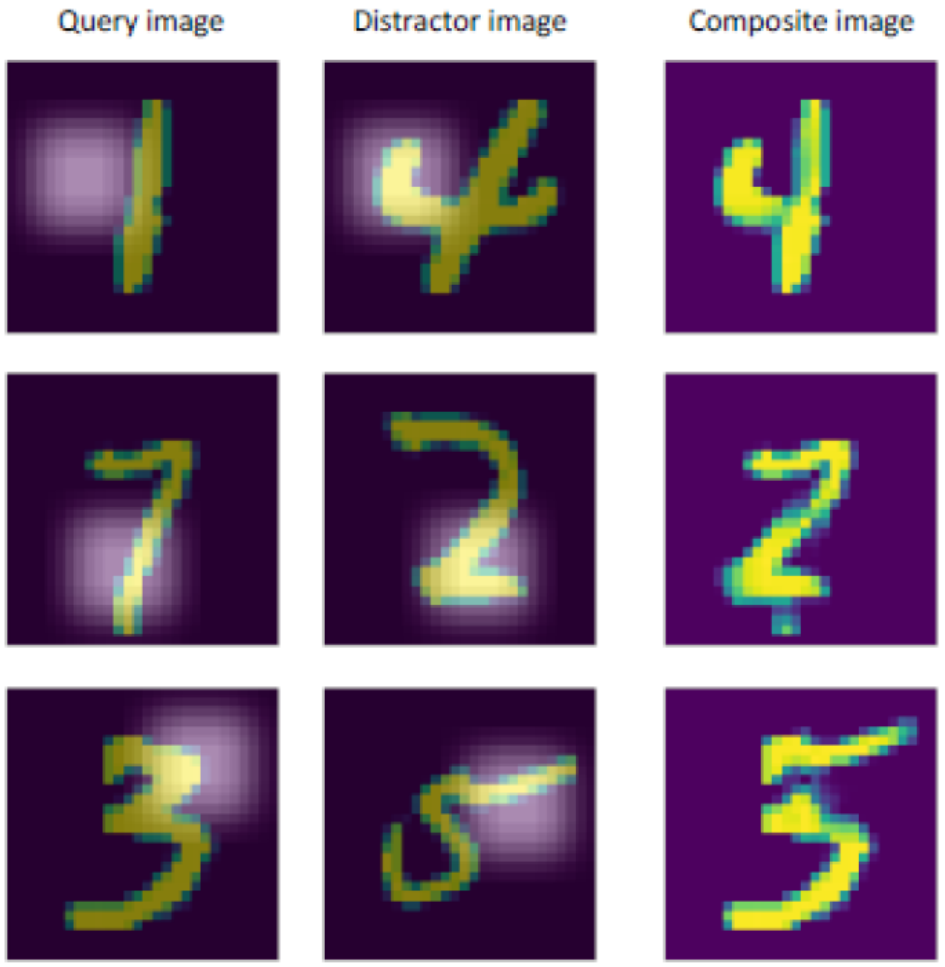
$$\text{s.t.} \quad c' = \text{argmax} \; g((\mathbb{1} - \mathbf{a}) \circ f(I) + \mathbf{a} \circ P f(I'))$$

$$a_i \in \{0, 1\} \;\; \forall i \;\; \text{and} \;\; P \in \mathcal{P}$$

- Too many choices for permutation matrix P and gating vector a
- Greedy strategy: Do edits one at a time
- Single edit: Find one feature out of hw possibilities in f(I) to be replaced with one feature in f(I')
- Choose the edit that maximizes the prediction of class c' for the edited image

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Evaluation of CNN on MNIST Dataset

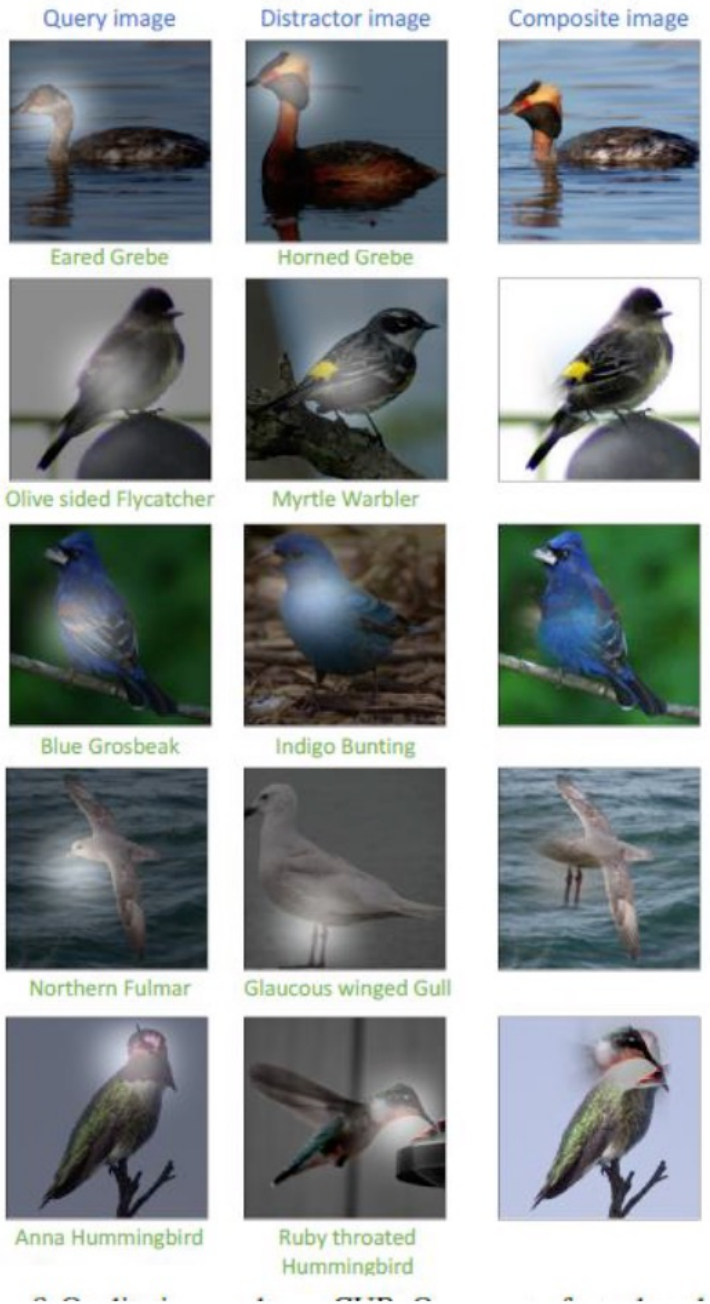


"The third column depicts a composite image generated in pixel space by aligning and superimposing the highlighted region centers."

19

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Evaluation of VGG-16 on Caltech-UCSD Birds Dataset



20

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000