# 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

# Sample Work from Our Research

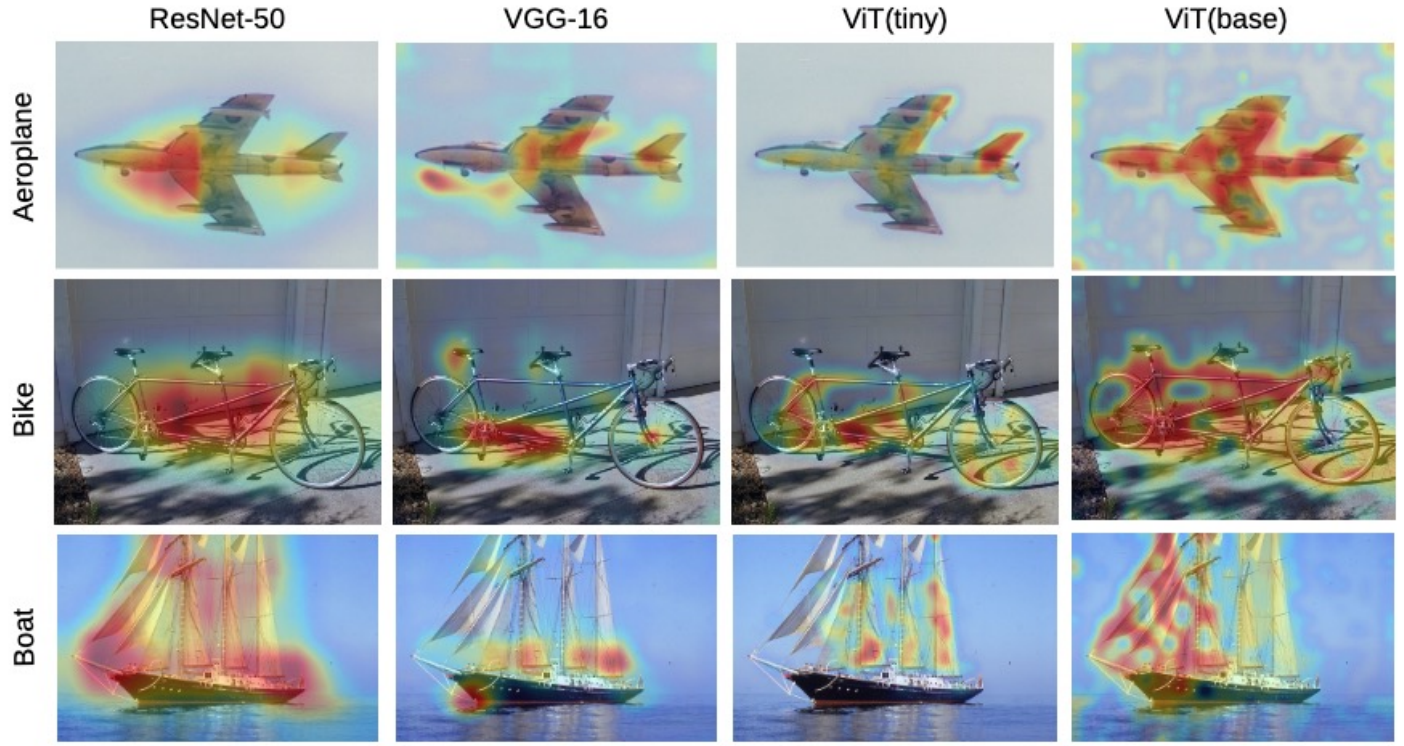


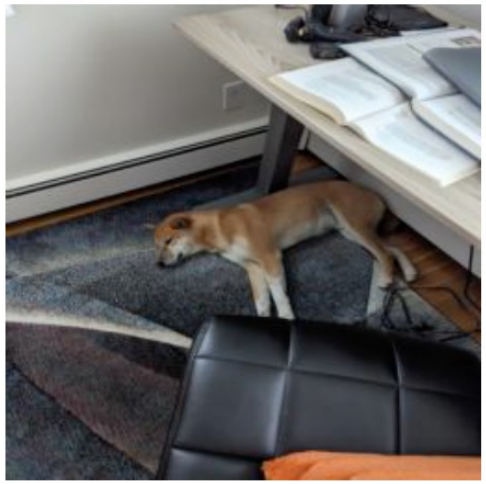

Figure 6: Exp. 1: Example heatmap visualizations. The visualization method is GradCam + SESS.

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.

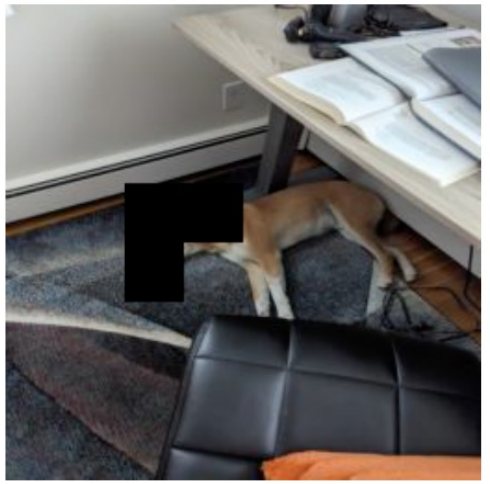
Previously on CENG7880

# Subtractive metrics

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



"Dog" (97%)



"Dog" (50%)

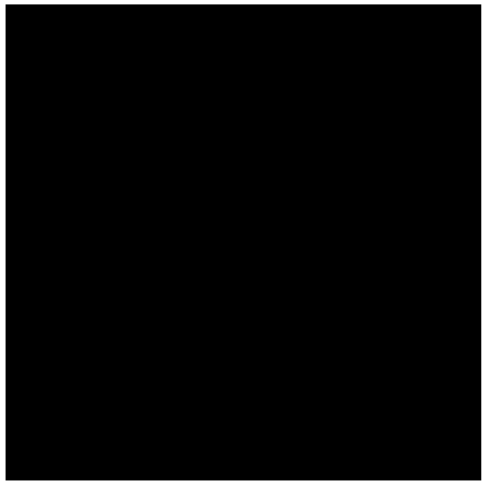
*I made up these numbers

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Previously on CENG7880

# 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

Previously on CENG7880

# Example of other metrics

Perturbation:

- How sensitive is your metric to perturbations?

Compactness:

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

Connectedness:

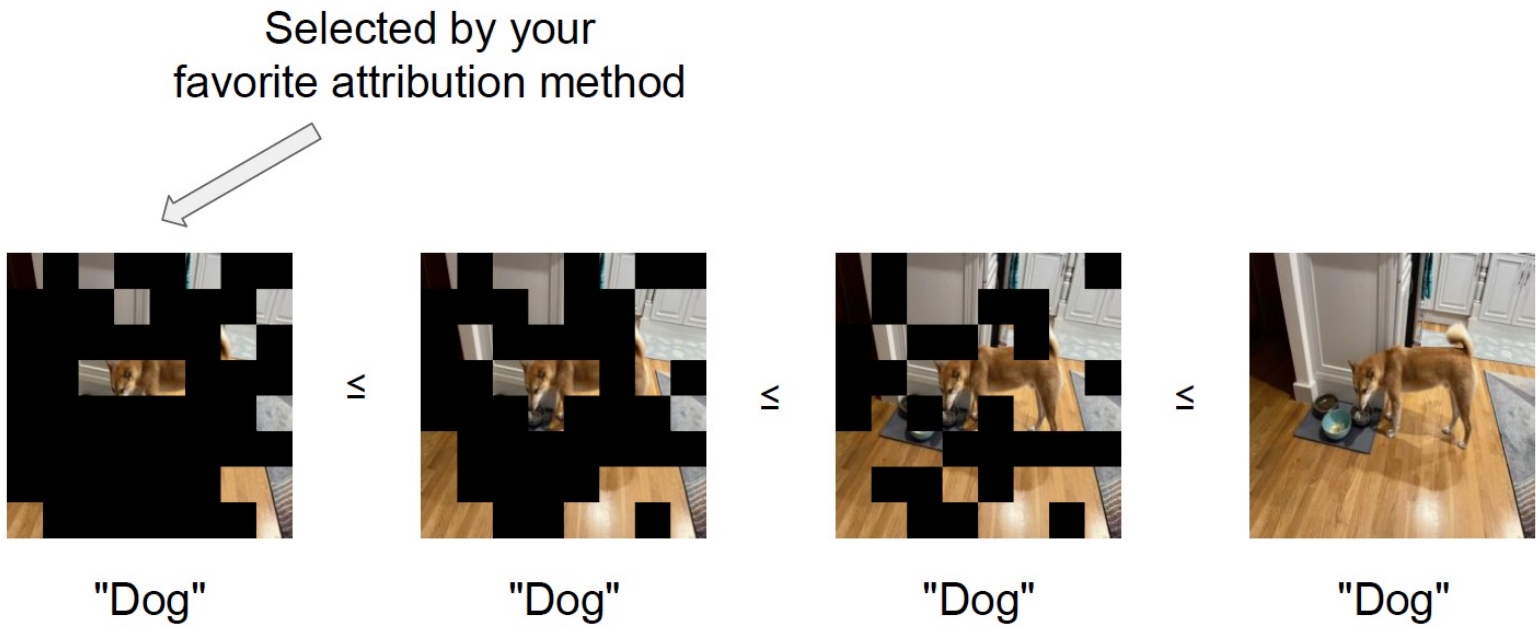
- Are two candidate explanations "connected" in some sense?

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

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Previously on CENG7880

# 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 $\alpha$ = BinaryAttribution(f,x)

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Previously on CENG7880

# Step 1: incremental stability

$f($  $) \cong f($  $+\Delta\ )$ for all small $\Delta$

Sufficient condition: Lipschitz wrt masking of features

- L1 norm on binary vectors = number of differences

$f($  $) - f($  $) \leq \lambda ||$  $-$  $||_1$ for all  , 

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

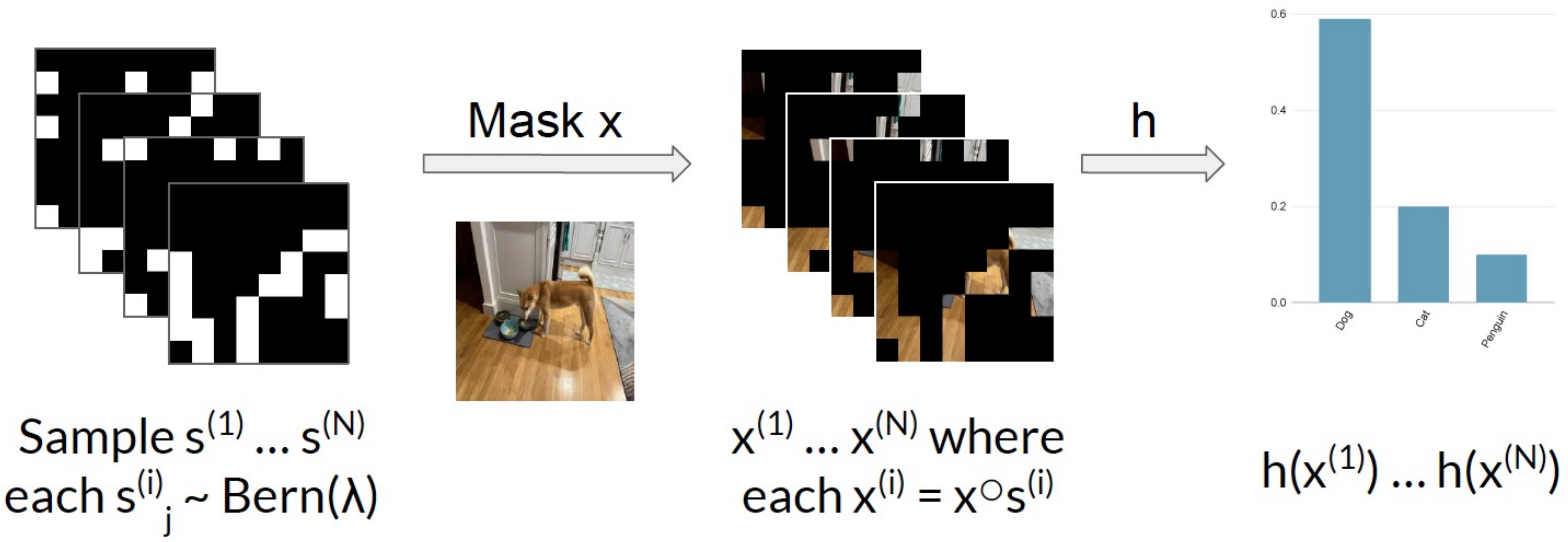
$$f(x \circ \alpha) - f(x \circ \alpha') \leq \lambda ||\alpha - \alpha'||_1 \quad \text{for all } \alpha, \alpha' \text{ in } \{0,1\}^n$$

Previously on CENG7880

# 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

Previously on CENG7880

# Step 3: provable incremental stability

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

Let $f(x) = MuS(h, x)$ with parameter $\lambda$

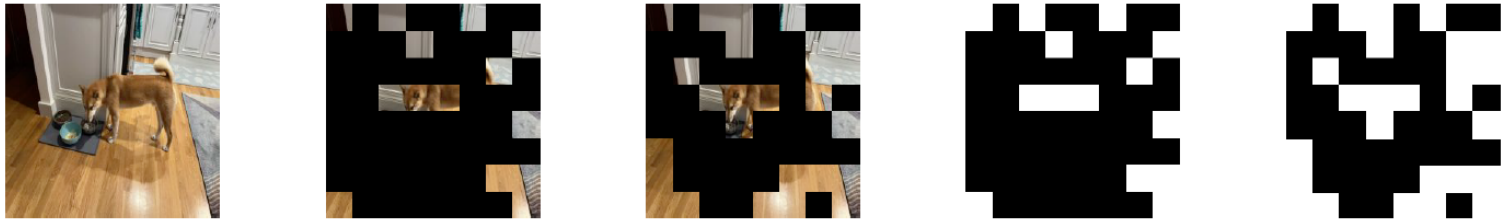
Probability of the Bernouilli distribution (in the previous slide)

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

**Theorem (MuS).** Suppose that

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

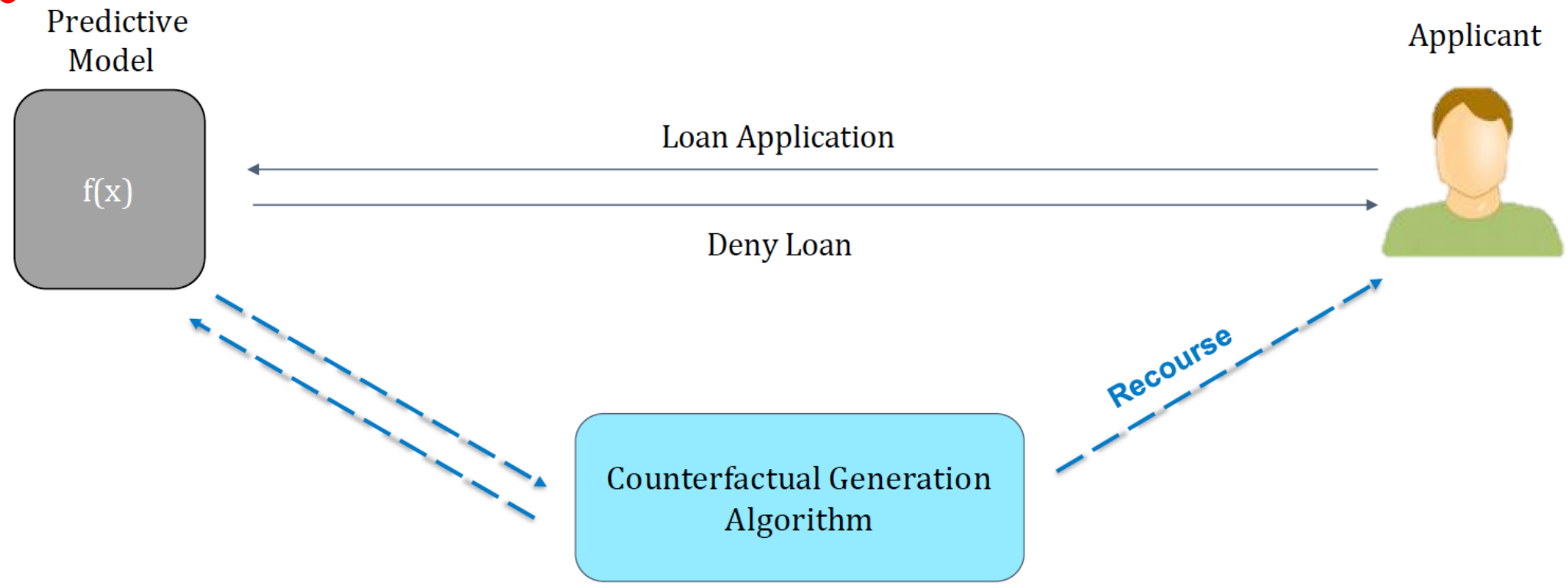
then for any $\alpha' \geq \alpha$ with $||\alpha' - \alpha||_1 \leq r$, we have $f(x \circ \alpha') \cong f(x \circ \alpha)$.



$x$ $x \circ \alpha$ $x \circ \alpha'$ $\alpha$ $\alpha'$

Previously on CENG7880

# Counterfactual Explanation



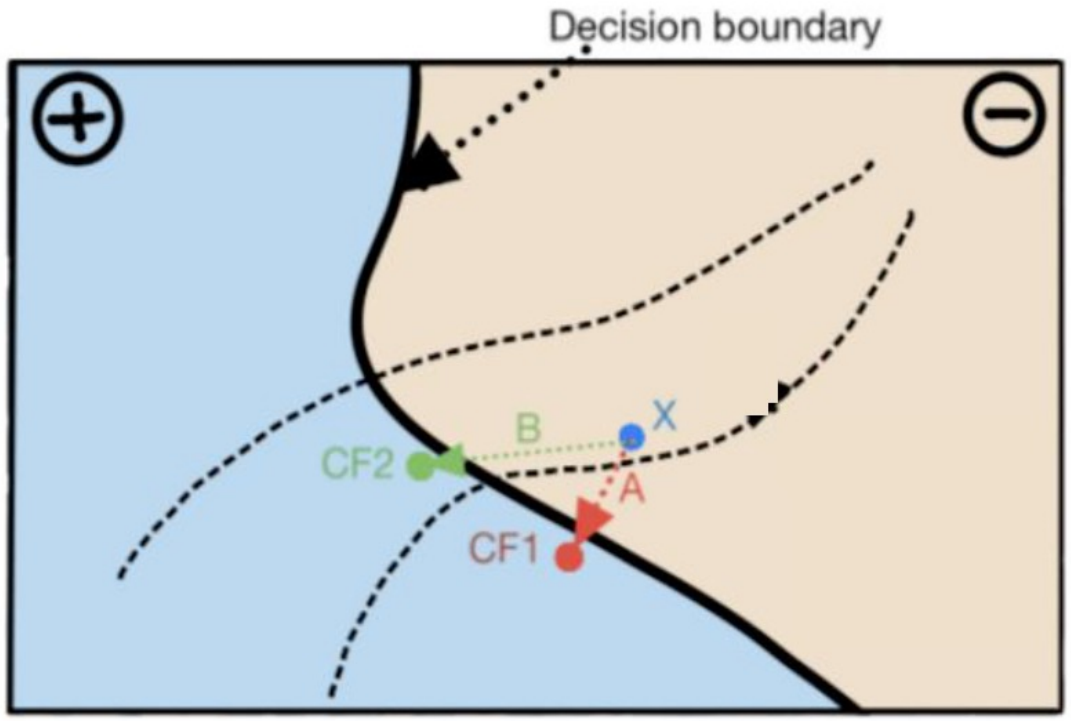

**Recourse**: Increase your salary by 5K & pay your credit card bills on time for next 3 months

4

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Previously on CENG7880

# Generating Counterfactual




Proposed solutions differ on:

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

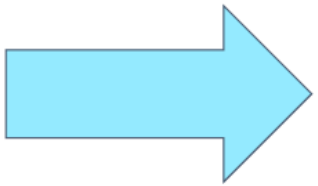
5

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Previously on CENG7880

# 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

Previously on CENG7880

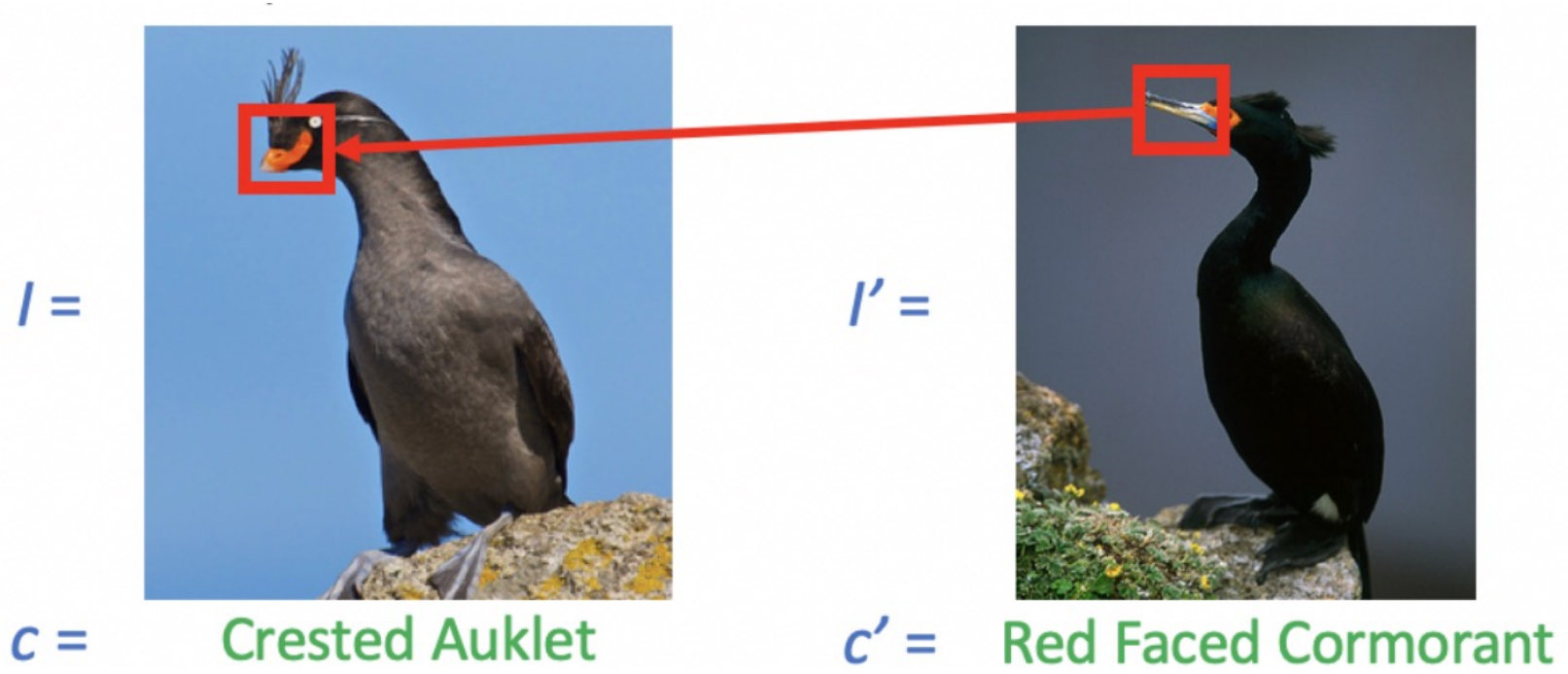



*Figure 1.* Our approach generates counterfactual visual explanations for a query image $I$ (left) – explaining why the example image was classified as class $c$ (*Crested Auklet*) rather than class $c'$ (*Red Faced Cormorant*) by finding a region in a distractor image $I'$ (right) and a region in the 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.

Previously on CENG7880

# Minimum-edit Counterfactual

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

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

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

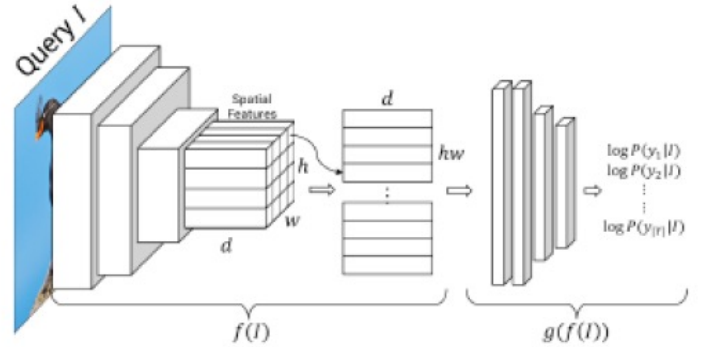



*Figure 3.* We decompose a CNN as a spatial feature extractor $f(I)$ and a decision network $g(f(I))$ as shown above.

*reshuffle features*

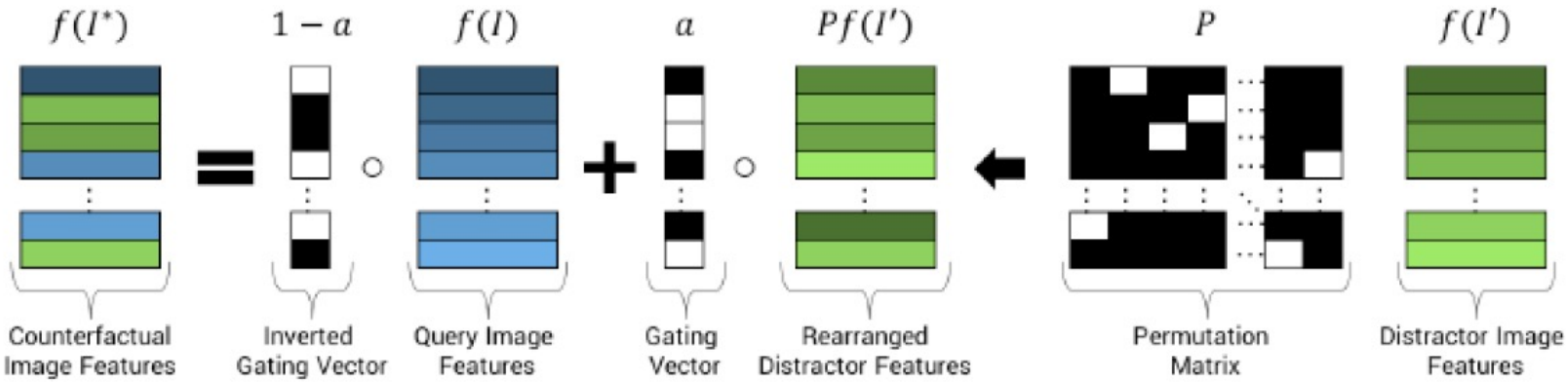
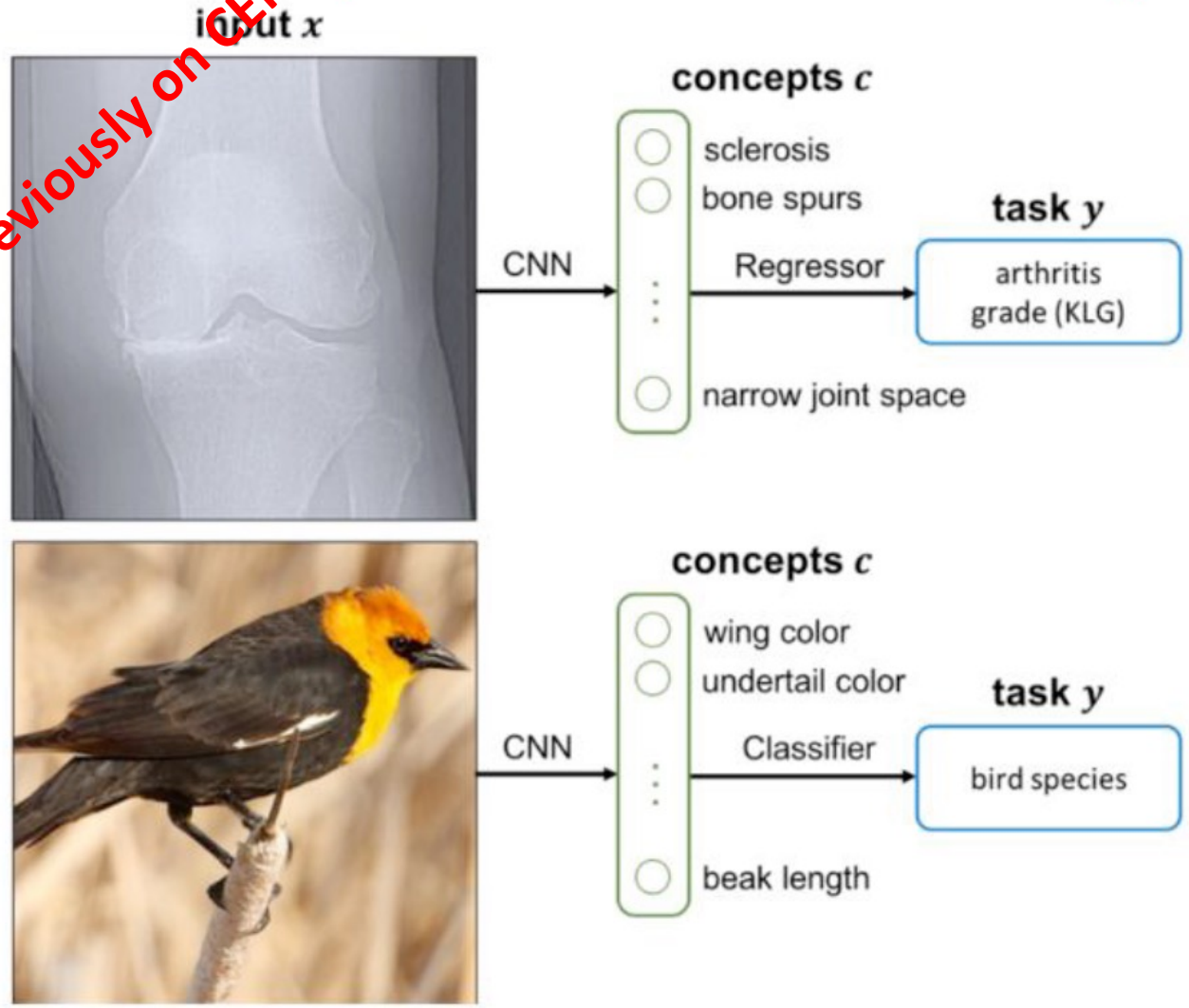



*Figure 2.* To parameterize our counterfactual explanations, we define a transformation that replaces regions in the query image $I$ with those from a distractor $I'$. Distractor image features $f(I')$ are first rearranged with a permutation matrix $P$ and then selectively replace entries in $f(I)$ according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

16

Previously on CENG7880

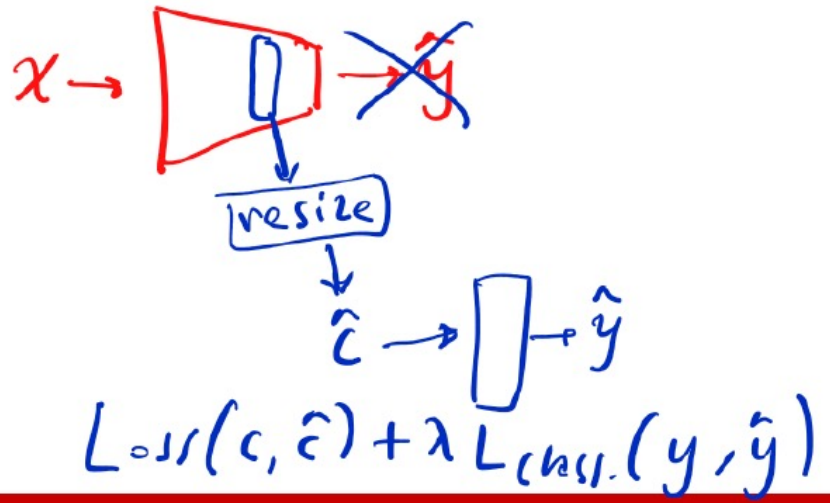
# Concept Bottleneck Model (CBM)




Koh et al. Concept bottleneck models. In ICML 2020.

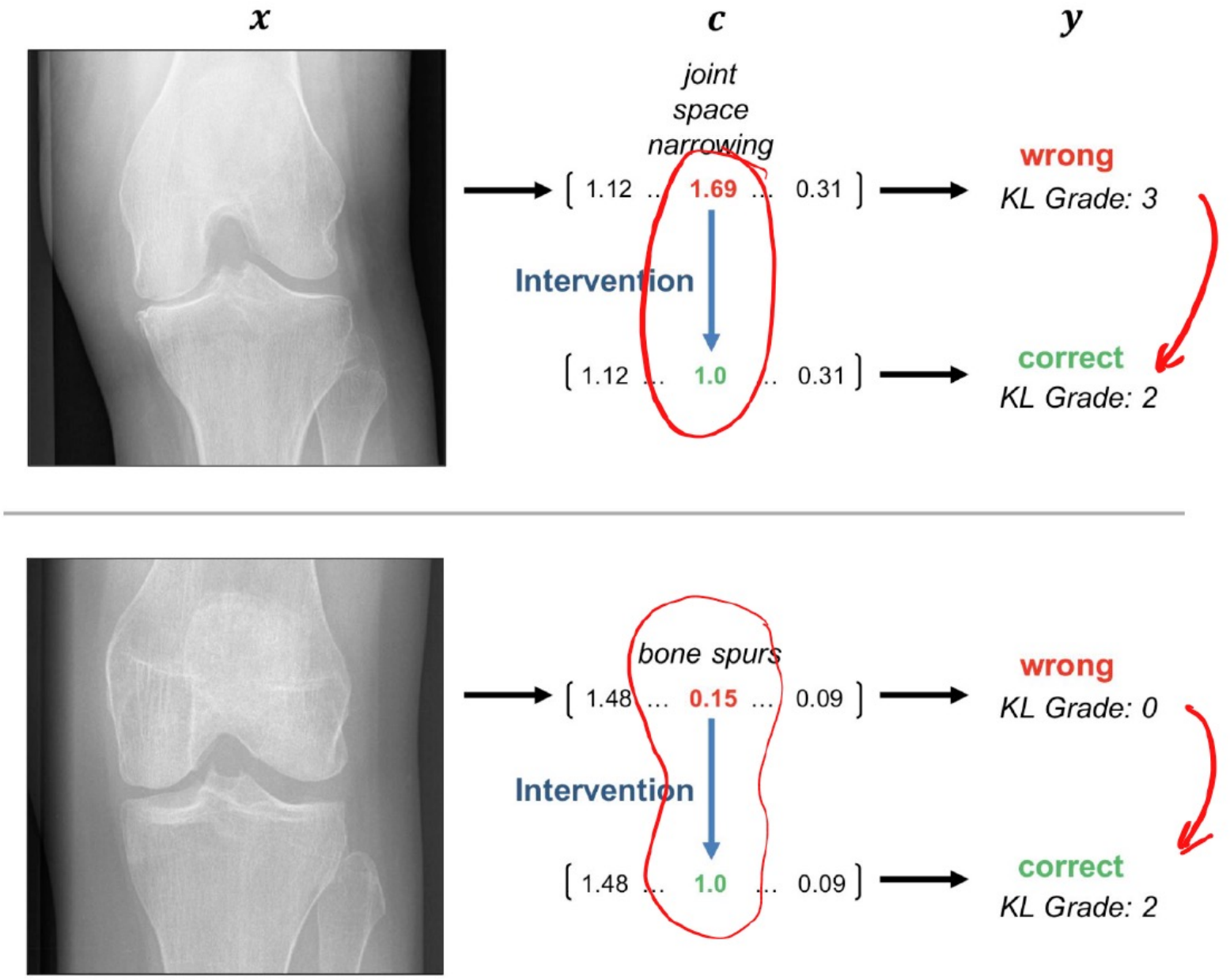
Attribute based classifiers since 2009
(see related work in Koh et al.)

Modern CBMs' promise is to convert any vision encoder to an explainable model.



Previously on CENG7880

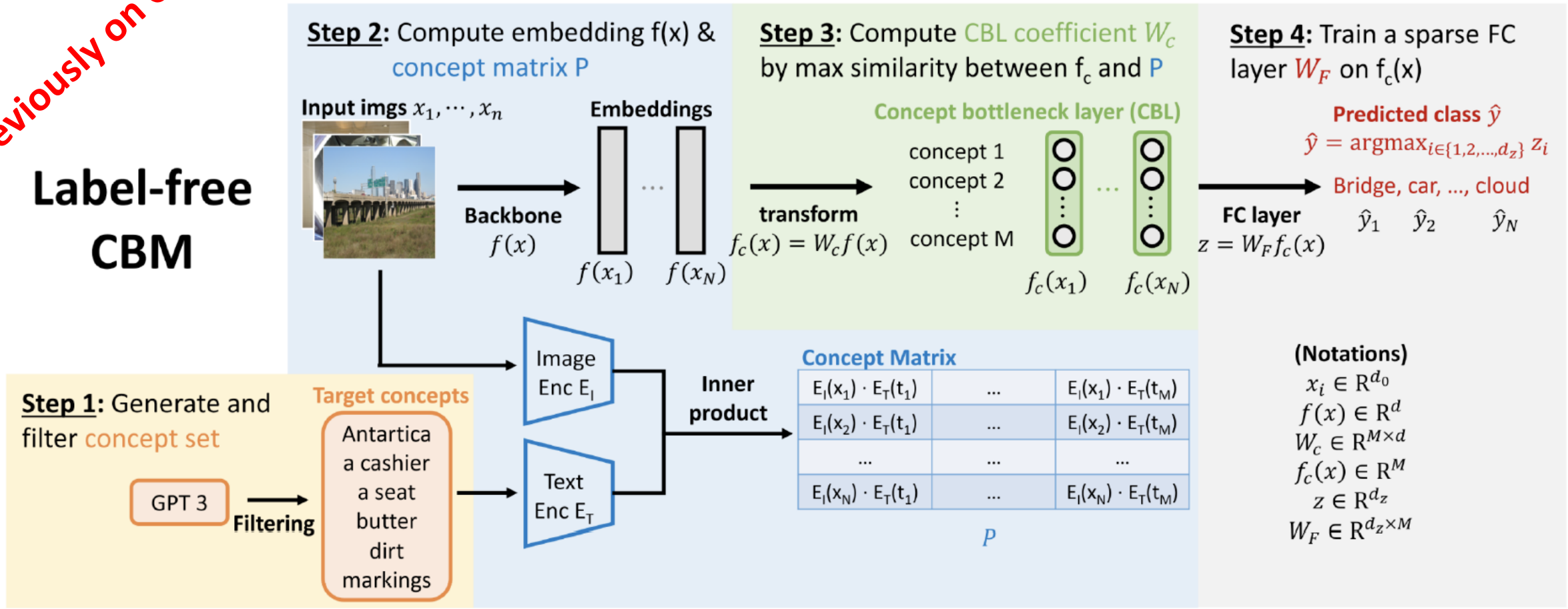
# Test-time intervention



# Label-free CBM (LFCBM)

Previously on CENG7880

## Label-free CBM




Oikarinen et al. Label-free concept bottleneck models. In ICLR 2023.

# Agenda

- Explainability
  - Representation-level explanations (Concept Activation Vectors)
  - Data attribution methods (Influence of training samples)
  - Explainability in LLMs

# Administrative Notes

- Final Exam:
  - 13 January 16:30

- Project milestones
  - **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

# Representation-based Explanations

# Local vs. Global Explanations

| | |
|---|---|
| Explain individual predictions | Explain complete behavior of the model |
| Help unearth biases in the *local neighborhood* of a given instance | Help shed light on *big picture biases* affecting larger subgroups |
| Help vet if individual predictions are being made for the right reasons | Help vet if the model, at a high level, is suitable for deployment |

21

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Representation-based Explanations

- Analyzing intermediate representations of a DNN can lead to model understanding
- Concept learning: Identify concepts that are semantically meaningful to humans and the model's reliance on such concepts

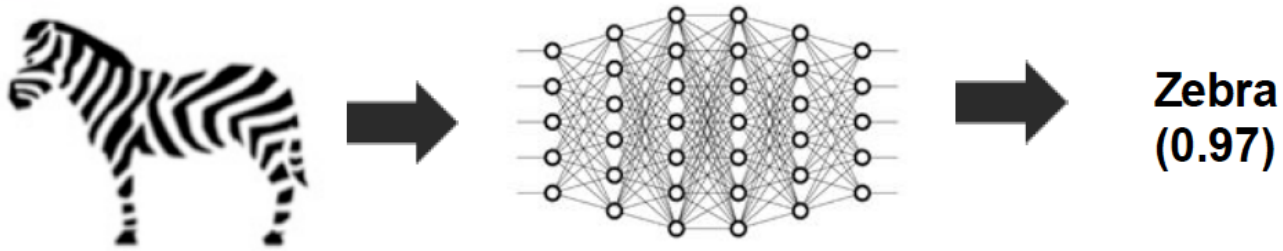
Interpretability beyond feature attribution: Quantitative testing with Concept Activation Vectors (TCAV)

Kim et al. ICML 2018



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Concept-based Explanations




How important is the notion of “stripes” for this prediction ?



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

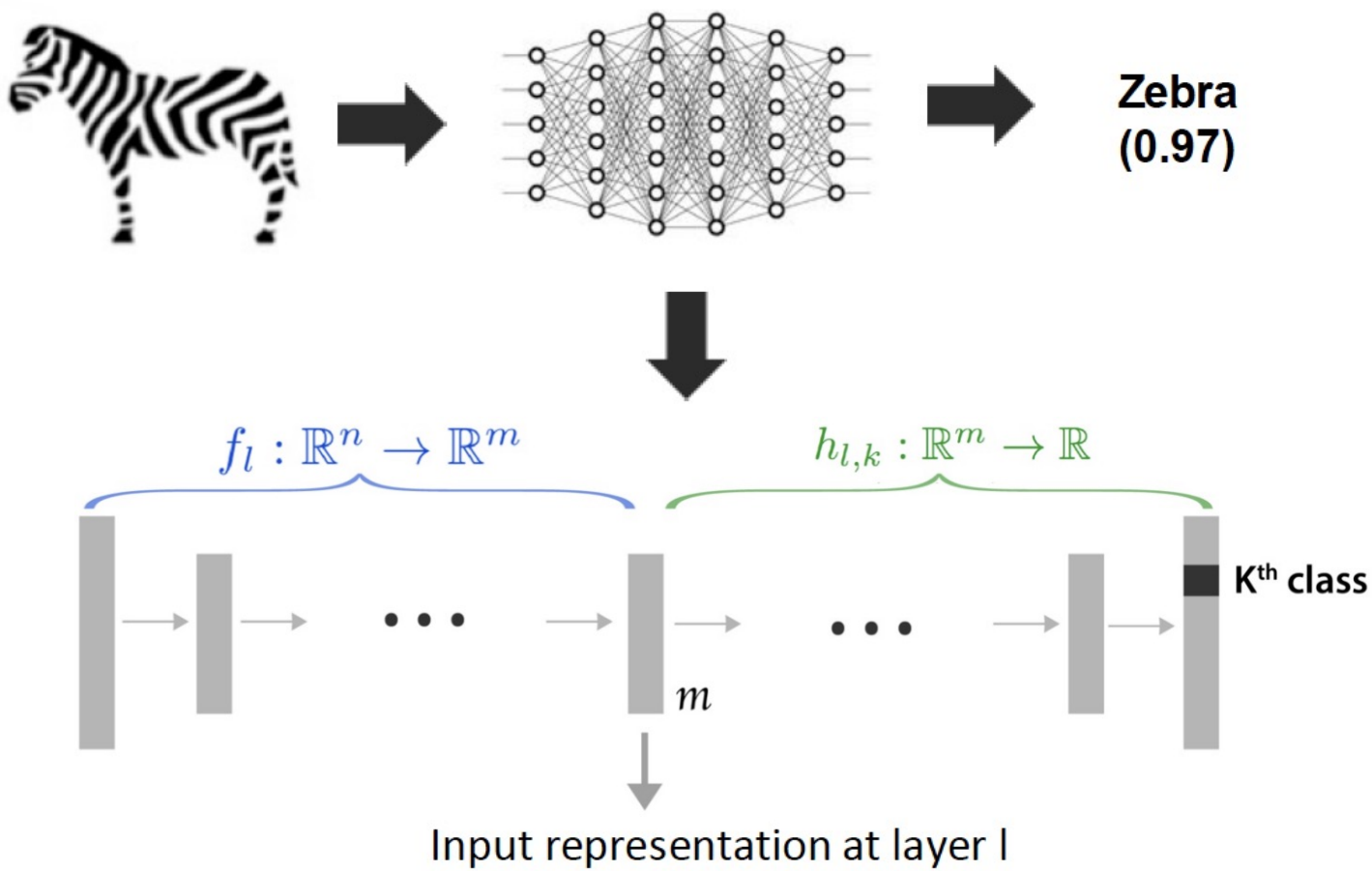
# Step 1: Specifying Concepts

- User needs to articulate the concepts of interest
- "Stripes" can be specified by giving positive examples
- Negative examples can be chosen randomly
  or given by the user



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

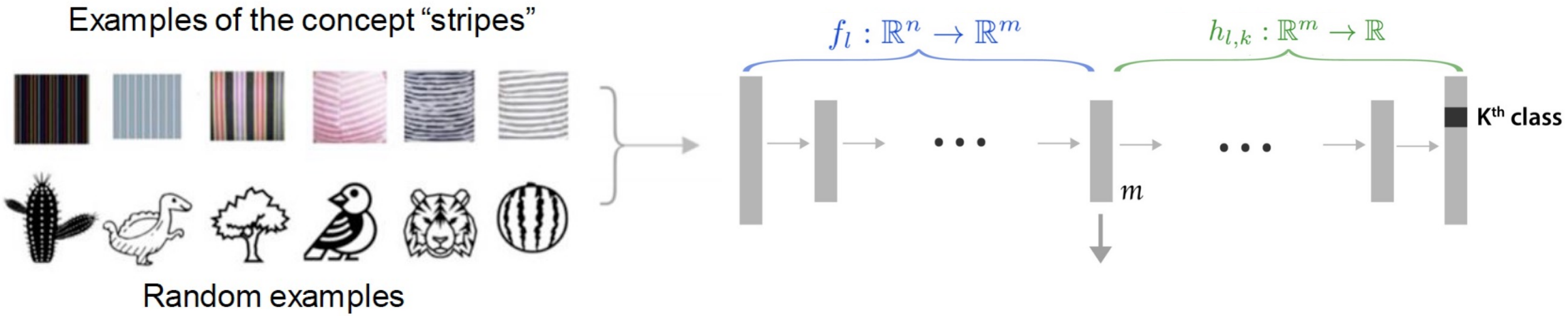
# Internal Layers in DNN



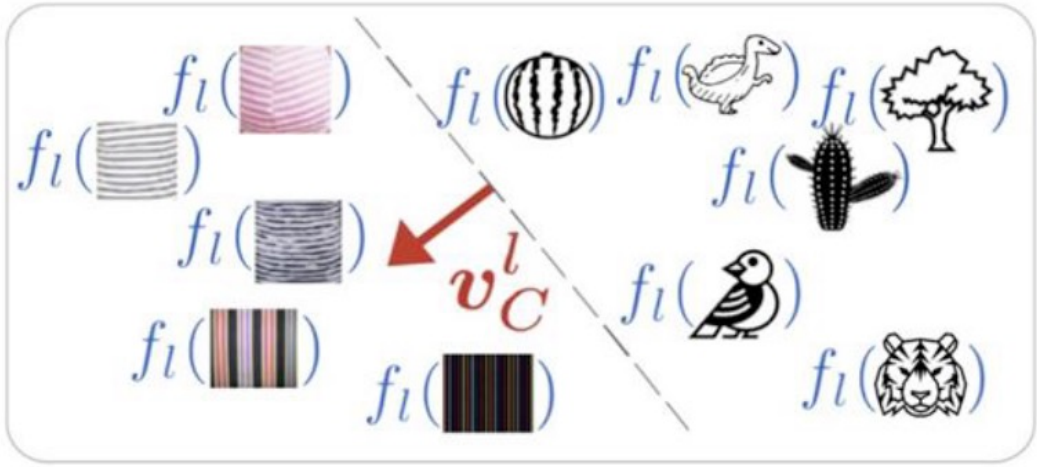

25

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Step 2: Compute Concept Activation Vector (CAV)




- Consider representations at layer l of all the positive and negative concept examples
- Train a linear classifier to separate positive from negative
- CAV: Vector orthogonal to the decision boundary






Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

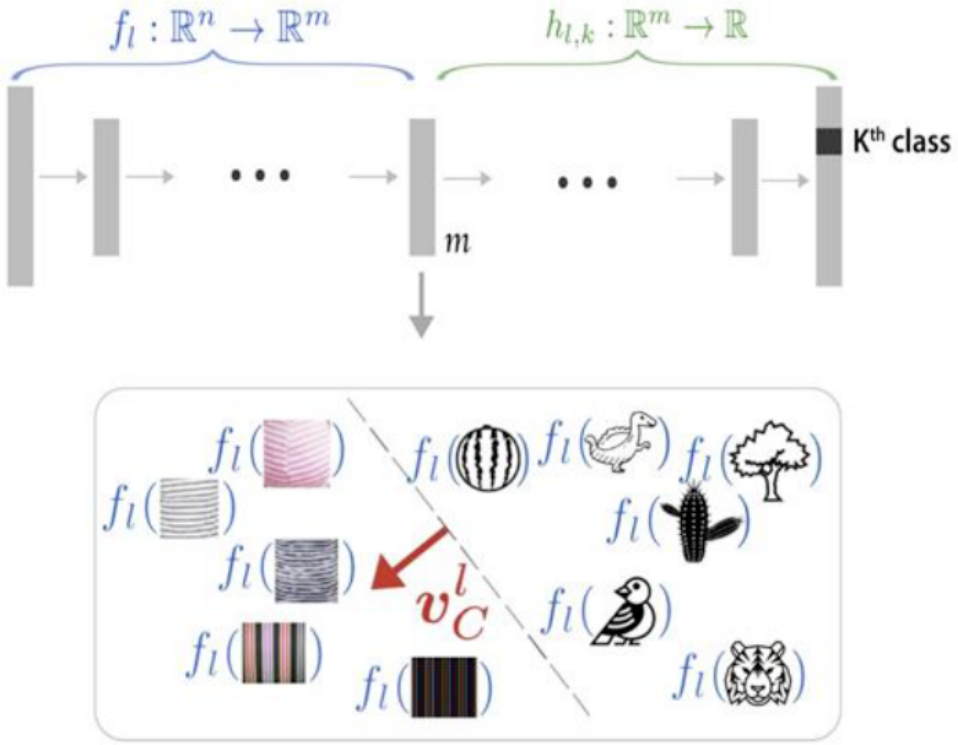
# Step 3: Compute Conceptual Sensitivity Score

- Recall: in saliency maps, sensitivity of output to an input feature/pixel is determined by the gradient w.r.t. input x

If $\boldsymbol{v}_C^l \in \mathbb{R}^m$ is a unit CAV vector for a concept $C$ in layer $l$, and $f_l(\boldsymbol{x})$ the activations for input $\boldsymbol{x}$ at layer $l$, the "conceptual sensitivity" of class $k$ to concept $C$ can be computed as the directional derivative $S_{C,k,l}(\boldsymbol{x})$:

$$\begin{aligned} S_{C,k,l}(\boldsymbol{x}) &= \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(\boldsymbol{x}) + \epsilon \boldsymbol{v}_C^l) - h_{l,k}(f_l(\boldsymbol{x}))}{\epsilon} \\ &= \nabla h_{l,k}(f_l(\boldsymbol{x})) \cdot \boldsymbol{v}_C^l, \end{aligned} \tag{1}$$

where $h_{l,k} : \mathbb{R}^m \to \mathbb{R}$. This $S_{C,k,l}(\boldsymbol{x})$ can quantitatively measure the sensitivity of model predictions with respect to concepts at any model layer. It is not a per-feature metric (e.g., unlike per-pixel saliency maps) but a per-concept scalar quantity computed on a whole input or sets of inputs.

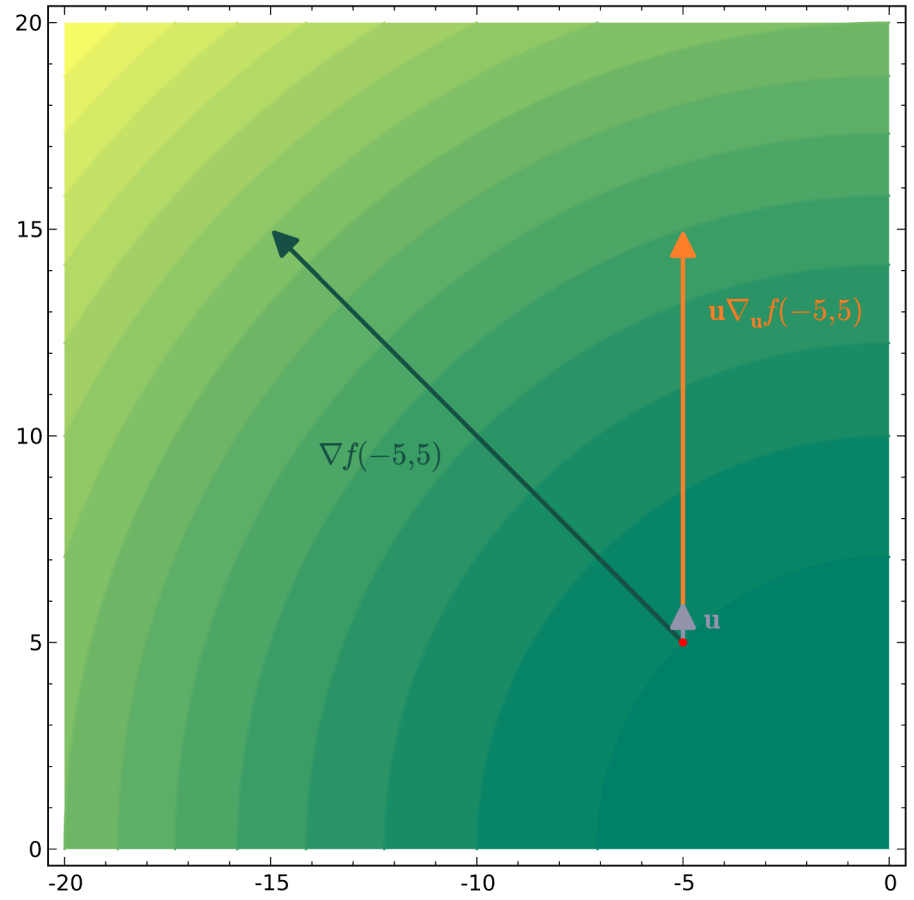





Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Background: Directional Derivative

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \frac{\mathbf{v}}{||\mathbf{v}||}$$

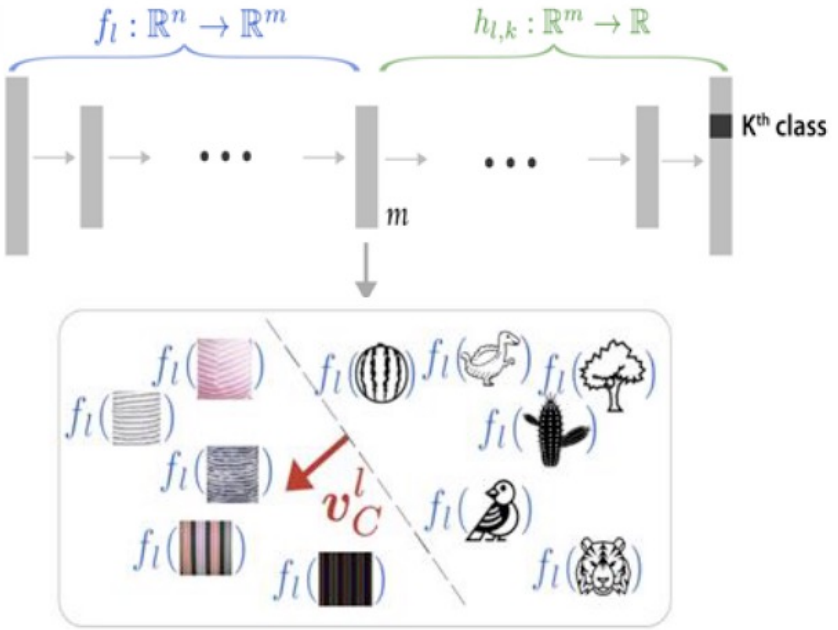



Material: Wikipedia

# Step 4: Testing with CAV (TCAV)

- Let $X_k$ be the set of all training inputs with label k

- Goal: Understand how a model f's prediction for class k is sensitive to a given concept C



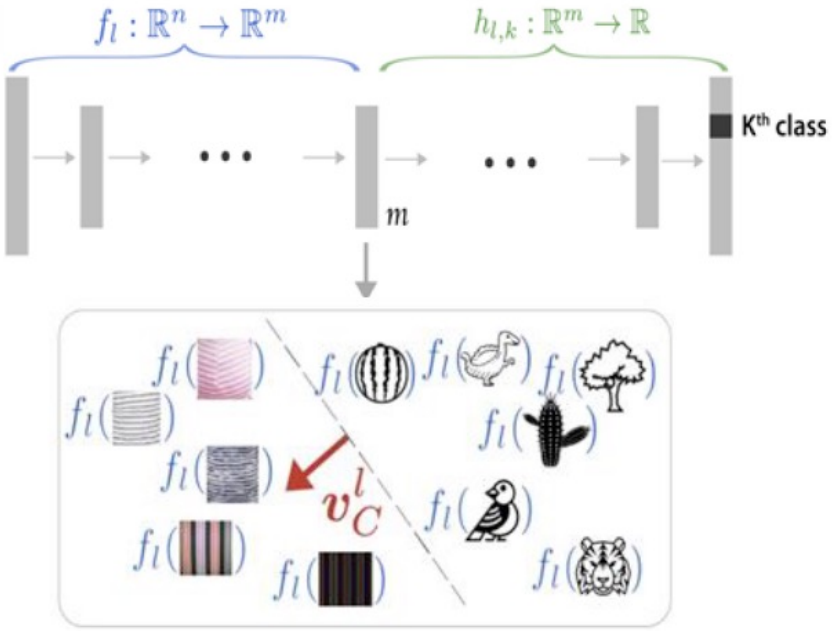

$$S_{C,k,l}(\boldsymbol{x}) = \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(\boldsymbol{x}) + \epsilon \boldsymbol{v}_C^l) - h_{l,k}(f_l(\boldsymbol{x}))}{\epsilon} = \nabla h_{l,k}(f_l(\boldsymbol{x})) \cdot \boldsymbol{v}_C^l, \quad (1)$$



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Step 4: Testing with CAV (TCAV)

- Let $X_k$ be the set of all training inputs with label k
- Goal: Understand how a model f's prediction for class k is sensitive to a given concept C
- TCAV score: Fraction of k-class training inputs whose l-layer activation vector was positively influenced by concept C



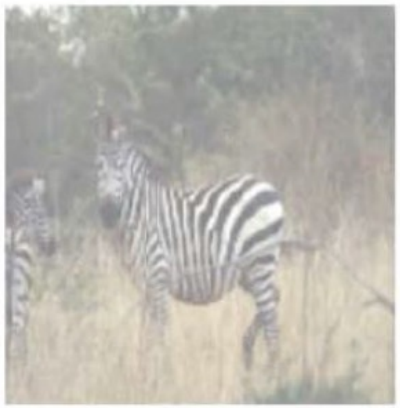

$$\text{TCAV}_{Q_{C,k,l}} = \frac{|\{\boldsymbol{x} \in X_k : S_{C,k,l}(\boldsymbol{x}) > 0\}|}{|X_k|}$$

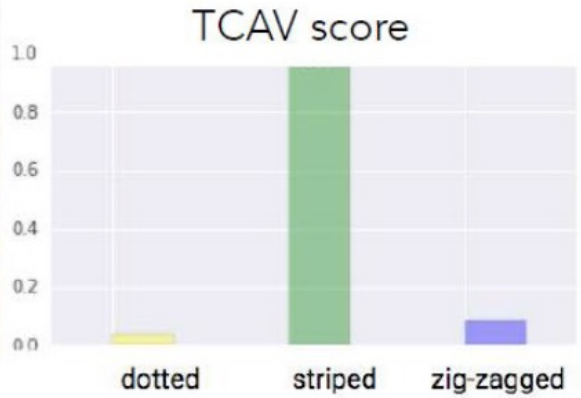
$$\begin{aligned} S_{C,k,l}(\boldsymbol{x}) &= \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(\boldsymbol{x}) + \epsilon \boldsymbol{v}_C^l) - h_{l,k}(f_l(\boldsymbol{x}))}{\epsilon} \\ &= \nabla h_{l,k}(f_l(\boldsymbol{x})) \cdot \boldsymbol{v}_C^l, \end{aligned} \quad (1)$$



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# TCAV Illustration






**zebra**-ness → $\frac{\partial p(z)}{\partial \boldsymbol{v}_C^l} = S_{C,k,l}(\boldsymbol{x})$

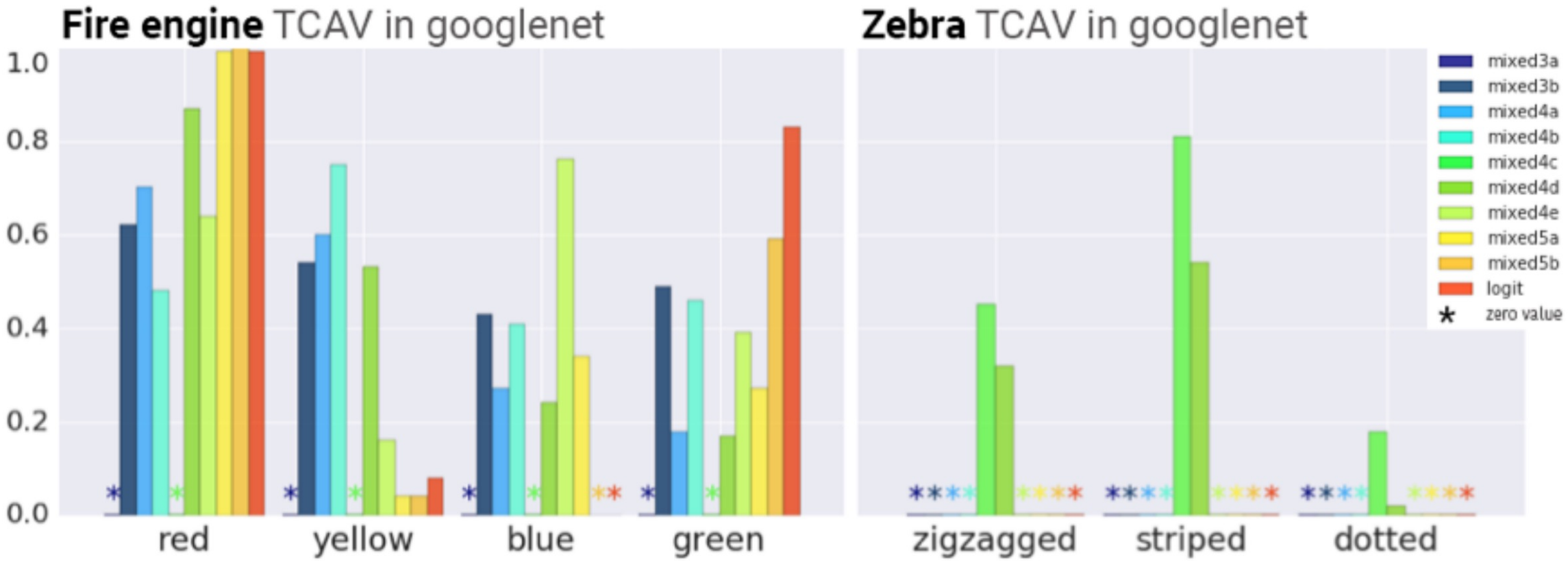
**striped** CAV →

**Directional derivative with CAV**



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# TCAV Evaluation

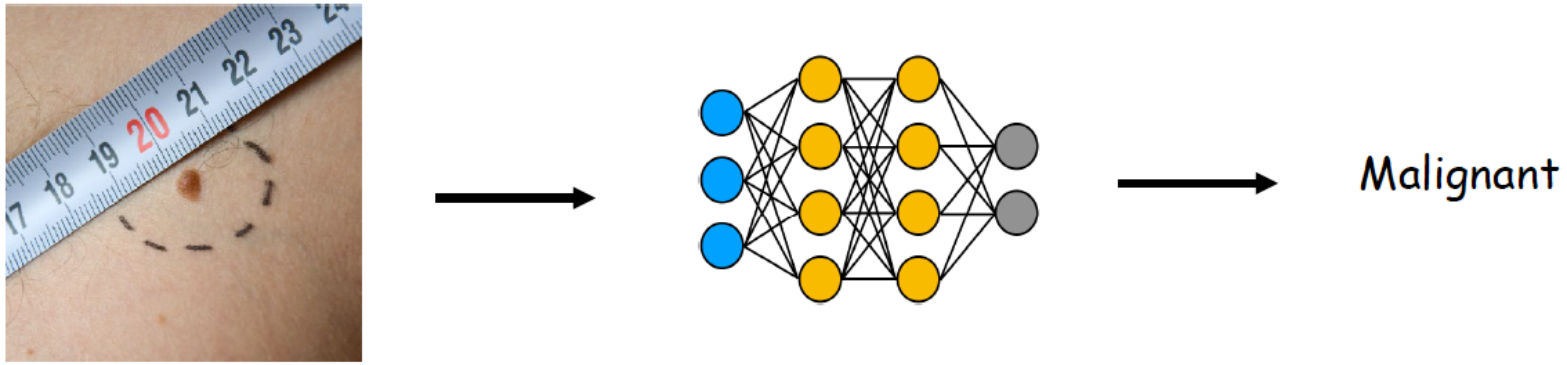



31

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Data Attribution Methods

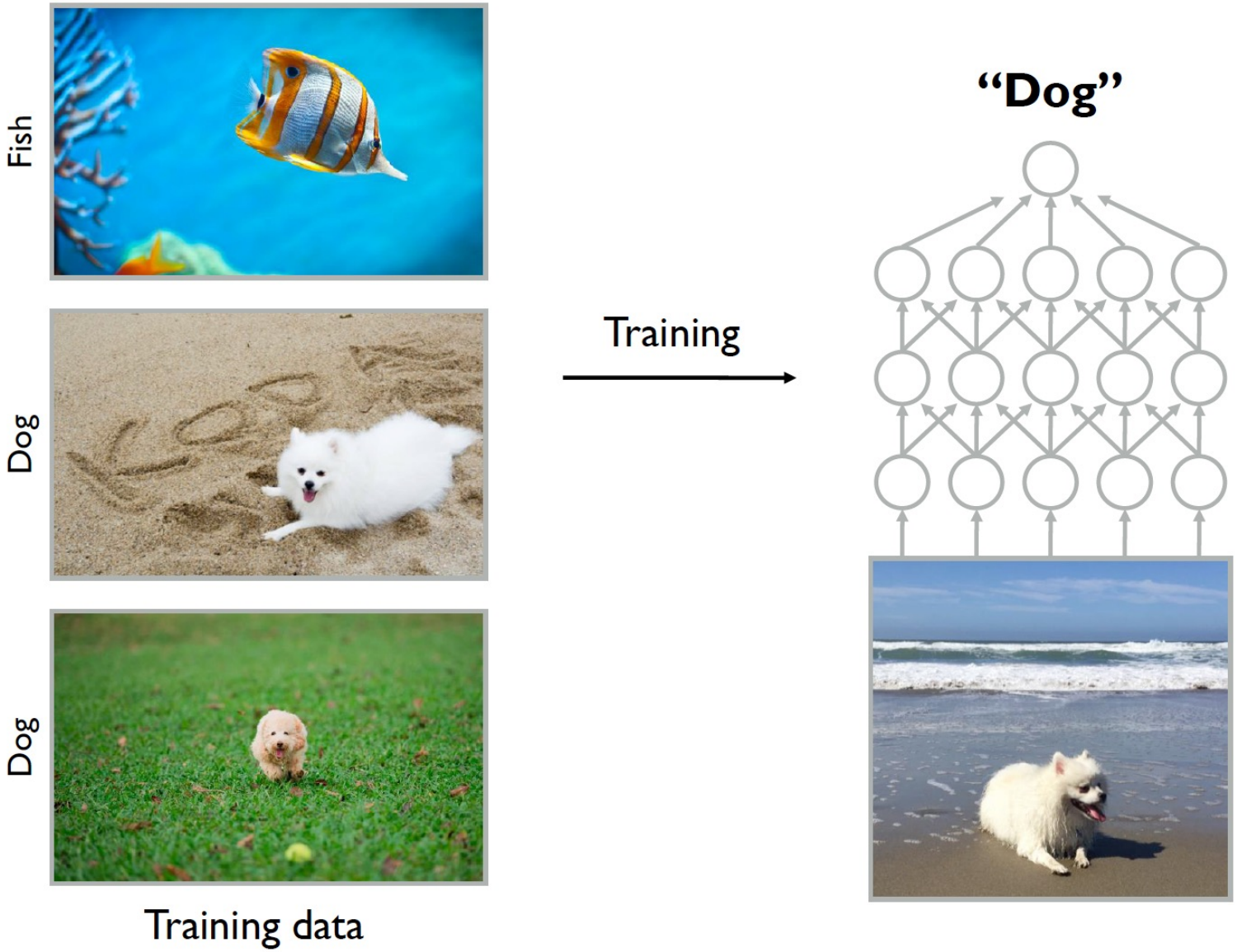
# Dat Attribution




Understand how the choice of training data influences the model's prediction

3

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000




Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Why did the model make this prediction?

↓

Which training points were most responsible for this prediction?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# The influence of individual training points

Koh & Liang, Understanding Black-box Predictions via Influence Functions, *ICML* 2017

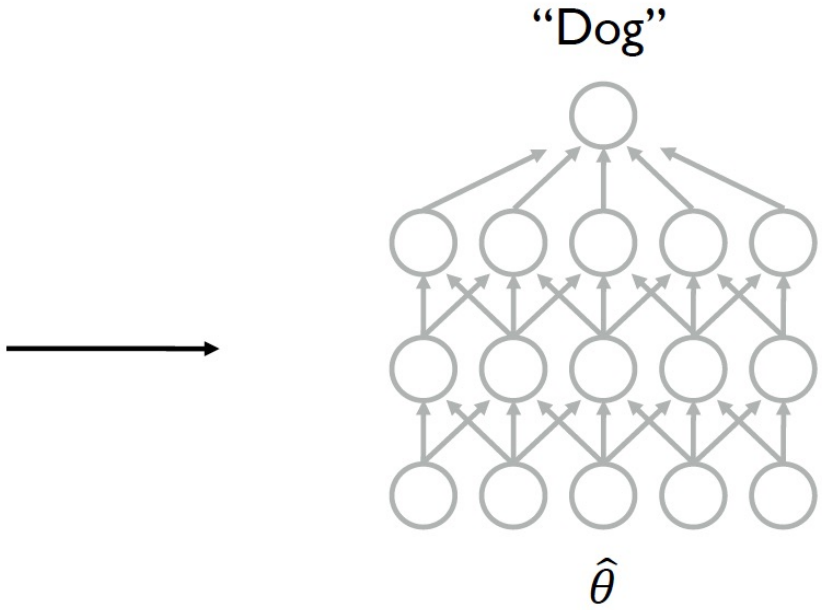
Counterfactual explanation: how would the model's predictions change if we did not have this training point?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Pick $\hat{\theta}$ to minimize $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$


$z_{train}$

Training data $z_1, z_2, \ldots, z_n$




Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000



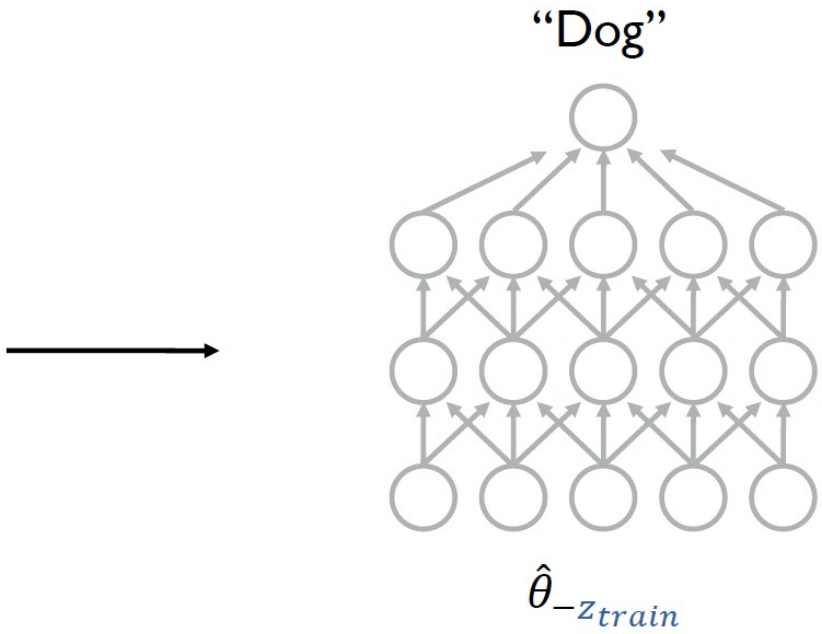
$z_{train}$


Training data $z_1, z_2, \ldots, z_n$

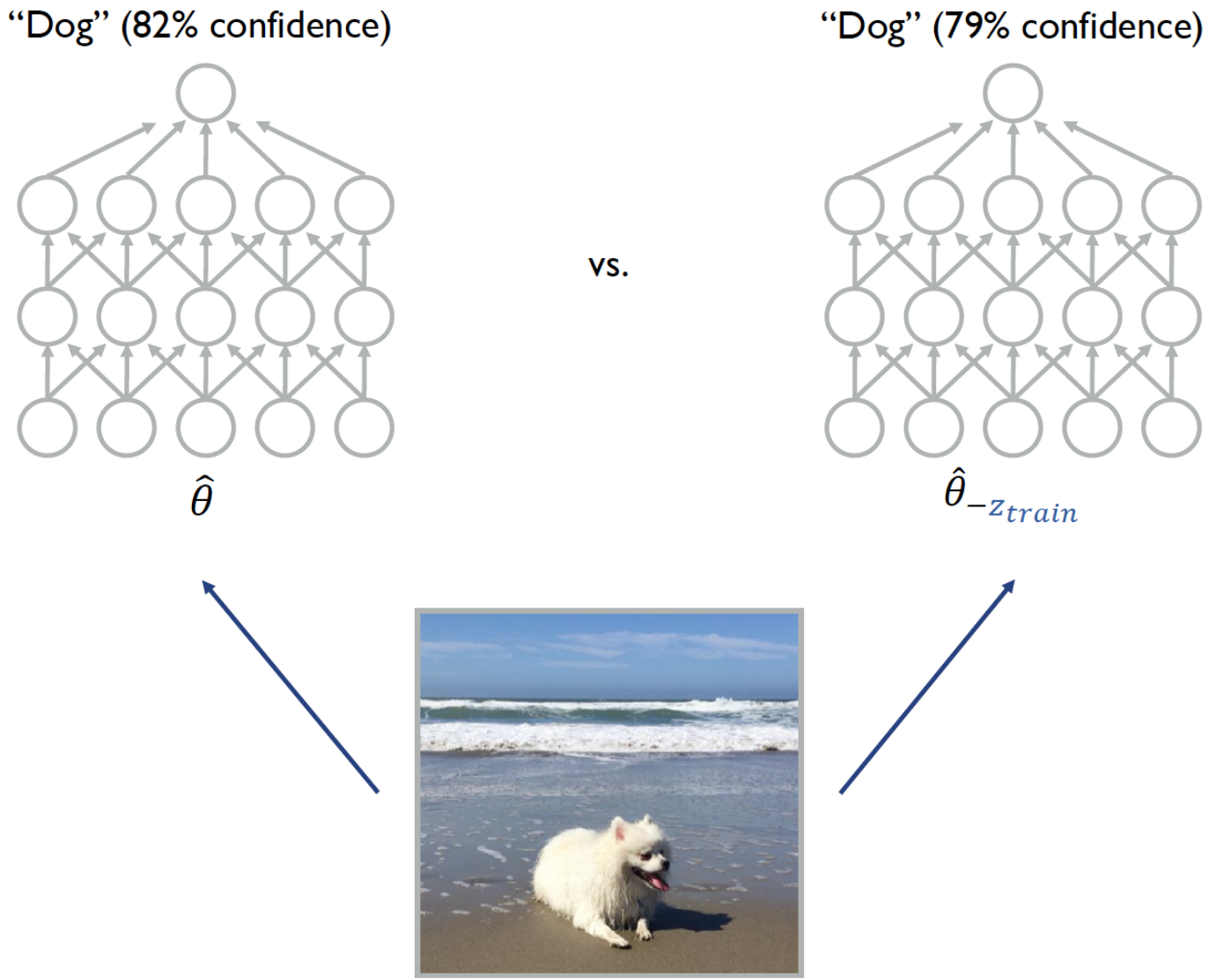
Pick $\hat{\theta}$ to minimize $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$

Pick $\hat{\theta}_{-z_{train}}$ to minimize $\frac{1}{n}\sum_{i \neq train}^{n} L(z_i, \theta)$

$$= \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta) - \frac{1}{n} L(z_{train}, \theta)$$

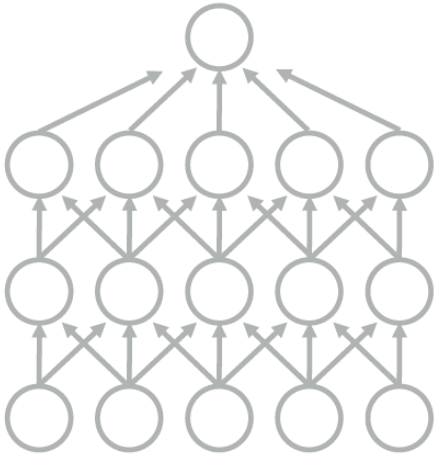



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000




Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

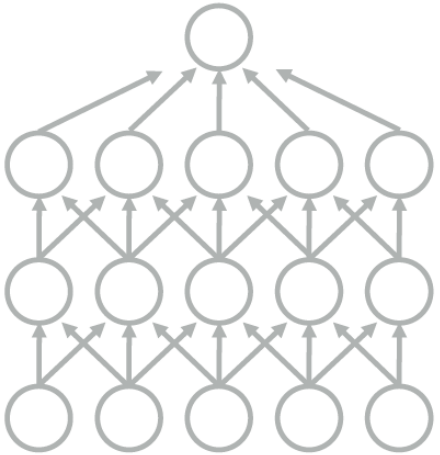
"Dog" (82% confidence)



$\hat{\theta}$

vs.

"Dog" (79% confidence)



$\hat{\theta}_{-z_{train}}$

What is $L(z_{test}, \hat{\theta}_{-z_{train}}) - L(z_{test}, \hat{\theta})$?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

| | |
|---|---|
| Problem | Repeatedly removing a training point and retraining the model is too slow |
| Solution | Approximation via influence functions (a classical technique from the 1970s) |

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Influence on the parameters

Calculate the impact of removing sample "$z$"

$$\frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) - \frac{1}{n}L(z,\theta)$$

Fortunately, influence functions give us an efficient approximation. The idea is to compute the parameter change if $z$ were upweighted by some small $\epsilon$, giving us new parameters $\hat{\theta}_{\epsilon,z} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z,\theta)$. A classic result (Cook & Weisberg, 1982) tells us that the influence of upweighting $z$ on the parameters $\hat{\theta}$ is given by

$$\mathcal{I}_{\text{up,params}}(z) \stackrel{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta}), \quad (1)$$

where $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta^2 L(z_i,\hat{\theta})$ is the Hessian and is positive definite (PD) by assumption. In essence, we are forming a quadratic approximation to the empirical risk around $\hat{\theta}$ and take a single Newton step; see appendix A for a derivation. Since removing a point $z$ is the same as upweighting it by $\epsilon = -\frac{1}{n}$, we can linearly approximate the parameter change due to removing $z$ without retraining the model by computing $\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n}\mathcal{I}_{\text{up,params}}(z)$.

Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning* (pp. 1885-1894). PMLR.

# Derivation of the gradient

f(x) is strongly convex:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^{\mathrm{T}}(\mathbf{y}-\mathbf{x}) + \frac{\gamma}{2}||\mathbf{x}-\mathbf{y}||_2^2$$

Recall that $\hat{\theta}$ minimizes the empirical risk:

$$R(\theta) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta). \tag{6}$$

We further assume that $R$ is twice-differentiable and strongly convex in $\theta$, i.e.,

$$H_{\hat{\theta}} \stackrel{\text{def}}{=} \nabla^2 R(\hat{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta}) \tag{7}$$

exists and is positive definite. This guarantees the existence of $H_{\hat{\theta}}^{-1}$, which we will use in the subsequent derivation.

The perturbed parameters $\hat{\theta}_{\epsilon,z}$ can be written as

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta\in\Theta} \{R(\theta) + \epsilon L(z,\theta)\}. \tag{8}$$

Define the parameter change $\Delta_\epsilon = \hat{\theta}_{\epsilon,z} - \hat{\theta}$, and note that, as $\hat{\theta}$ doesn't depend on $\epsilon$, the quantity we seek to compute can be written in terms of it:

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon}. \tag{9}$$

Since $\hat{\theta}_{\epsilon,z}$ is a minimizer of (8), let us examine its first-order optimality conditions:

$$0 = \nabla R(\hat{\theta}_{\epsilon,z}) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z}). \tag{10}$$

Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning* (pp. 1885-1894). PMLR.

# Derivation of the gradient

Recall first-order Taylor series:

$$f(x) \approx f(a) + f'(a) \times (x - a)$$

$f(x)$ is the following term:

$$\nabla R(\hat{\theta}_{\epsilon,z}) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z})$$

with $x$: $\hat{\theta}_{\epsilon,z}$ and $a$: $\hat{\theta}$

Next, since $\hat{\theta}_{\epsilon,z} \to \hat{\theta}$ as $\epsilon \to 0$, we perform a Taylor expansion of the right-hand side:

$$0 \approx \left[\nabla R(\hat{\theta}) + \epsilon \nabla L(z, \hat{\theta})\right] + \left[\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z, \hat{\theta})\right] \Delta_\epsilon, \tag{11}$$

where we have dropped $o(\|\Delta_\epsilon\|)$ terms.

Solving for $\Delta_\epsilon$, we get:

$$\Delta_\epsilon \approx - \left[\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z, \hat{\theta})\right]^{-1} \left[\nabla R(\hat{\theta}) + \epsilon \nabla L(z, \hat{\theta})\right]. \tag{12}$$

Since $\hat{\theta}$ minimizes $R$, we have $\nabla R(\hat{\theta}) = 0$. Dropping $o(\epsilon)$ terms, we have

$$\Delta_\epsilon \approx - \nabla^2 R(\hat{\theta})^{-1} \nabla L(z, \hat{\theta}) \epsilon. \tag{13}$$

Combining with (7) and (9), we conclude that:

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla L(z, \hat{\theta}) \tag{14}$$

$$\stackrel{\text{def}}{=} \mathcal{I}_{\text{up,params}}(z). \tag{15}$$

Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning* (pp. 1885-1894). PMLR.

# Influence functions

- $\hat{\theta}_{\epsilon,z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z_{train},\theta)$
- Under smoothness assumptions,

$$I_{up,loss}(z_{train}, z_{test}) \stackrel{\text{def}}{=} \left.\frac{dL\left(z_{test}, \hat{\theta}_{\epsilon,z_{train}}\right)}{d\epsilon}\right|_{\epsilon=0}$$

Influence of removing $z_{train}$ on $z_{test}$

$$= \nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T \left.\frac{d\hat{\theta}_{\epsilon,z_{train}}}{d\epsilon}\right|_{\epsilon=0}$$

$$= -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z_{train}, \hat{\theta})$$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Influence functions

- $\hat{\theta}_{\epsilon, z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z_{train}, \theta)$
- Under smoothness assumptions,

$$I_{up,loss}(z_{train}, z_{test}) \stackrel{\text{def}}{=} \left. \frac{dL\left(z_{test}, \hat{\theta}_{\epsilon, z_{train}}\right)}{d\epsilon} \right|_{\epsilon=0}$$

$$= -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{train}, \hat{\theta}\right)^\top$$

where $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 L\left(z_i, \hat{\theta}\right)$.

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Influence functions

- $\hat{\theta}_{\epsilon,z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z_{train},\theta)$
- Under smoothness assumptions,

$$I_{up,loss}(z_{train}, z_{test}) \stackrel{\text{def}}{=} \left.\frac{dL\left(z_{test}, \hat{\theta}_{\epsilon,z_{train}}\right)}{d\epsilon}\right|_{\epsilon=0}$$

$$= -\nabla_\theta L\left(z_{test},\hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{train},\hat{\theta}\right)^\top$$

where $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L\left(z_i,\hat{\theta}\right)$.

- $L\left(z_{test},\hat{\theta}_{-z_{train}}\right) - L\left(z_{test},\hat{\theta}\right) = -\frac{1}{n} I_{up,loss}(z_{train}, z_{test})$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Debugging Models with Influence Functions

- Task: Image Classification
- Model 1: Support Vector Machine (SVM) with Radical Basis Function (RBF) kernel
- Model 2: Inception v3 network from CNN family
- Training dataset: ImageNet

5

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Debugging Models with Influence Functions

- Task: Image Classification
- Model 1: Support Vector Machine (SVM) with Radical Basis Function (RBF) kernel
- Model 2: Inception v3 network from CNN family
- Training dataset: ImageNet
- Sample correct prediction by both models: Fish

- Question: Which training images were most influential in the model's prediction?

6

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Debugging Models with Influence Functions

**RBF SVM**

**Inception**

4

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# Applications of Influence Functions

- Understanding model predictions
- Adversarial training examples
- Debugging domain mismatch (i.e. distribution shift in test-data vs. training-data)
- Fixing mislabeled examples

10

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

# The influence of groups of training points

Koh*, Ang*, Teo*, & Liang, On the Accuracy of Influence Functions for Measuring Group Effects [under review]

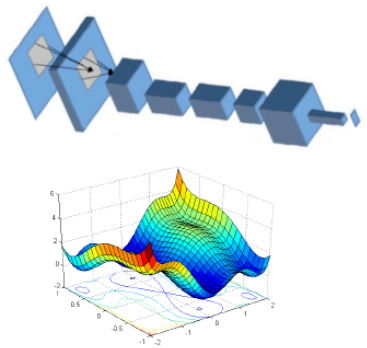
NeurIPS 2019

# Anatomy of an ML Prediction

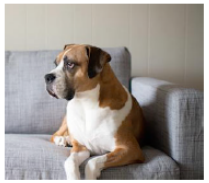
Training set $S$

+

Learning algorithm



Test input $x$



⟶ **"dog" (85%)**

Model output

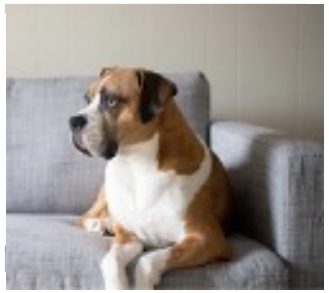
**Question:** How do training data and learning algorithms combine to yield model outputs?

# **Datamodels:** Data-to-Output Modeling

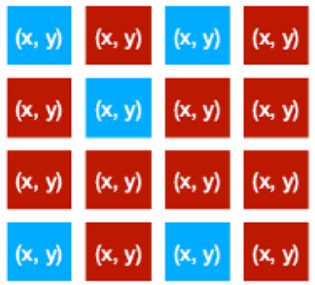
**What we are trying to compute (model output function):**

Output of interest on $x$
(think: margin of correct class)
after training on $S'$

$$f(x, S')$$



Specific input $x$




Subset $S'$ of the training set $S$

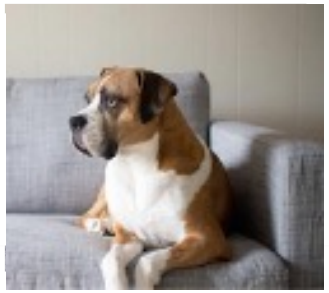
# **Datamodels:** Data-to-Output Modeling

**What we are trying to compute (model output function):**

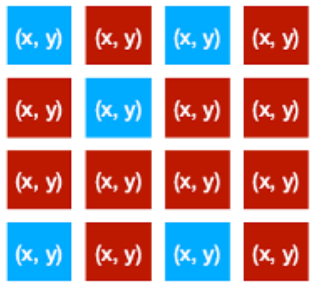
Output of interest on $x$
(think: margin of correct class)
after training on $S'$

$$f(x, S') \approx \hat{f}(x, S')$$

Datamodel for $x$



Specific input $x$




Subset $S'$ of the training set $S$

# **Datamodels:** Data-to-Output Modeling

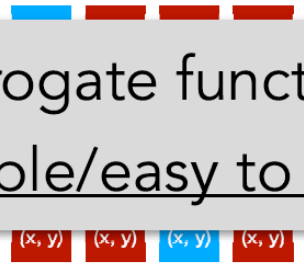
**What we are trying to compute (model output function):**

Output of interest on $x$
(think: margin of correct class)
after training on $S'$

$$f(x, S') \approx \hat{f}(x, S')$$

Datamodel for $x$

Specific input $x$




Subset $S'$ of the training set $S$

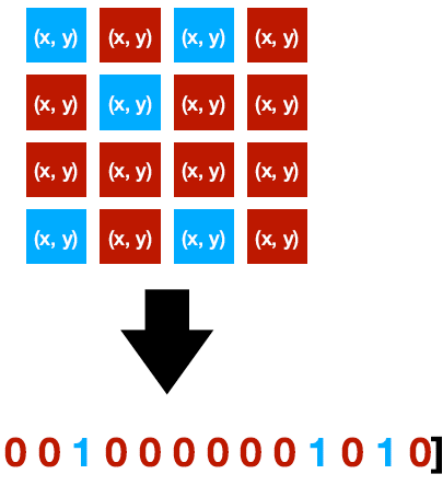
**Datamodeling framework**: Find a surrogate function $\hat{f}$ that approximates $f$, while also being simple/easy to analyze

# **Model Choice:** Linear

$$\hat{f}(x, S') = \theta_x^\top \mathbf{1}_{S'}$$

**Learned parameter:** vector of weights (one weight per training example in $S$)

Indicator vector of $S'$

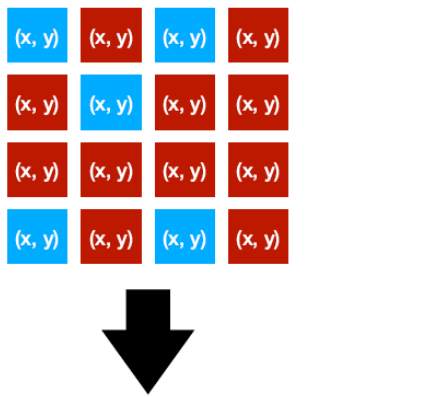


# **Model Choice:** Linear

$$\hat{f}(x, S') = \theta_x^\top \mathbf{1}_{S'}$$

**Learned parameter:** vector of weights (one weight per training example in $S$)

Indicator vector of $S'$

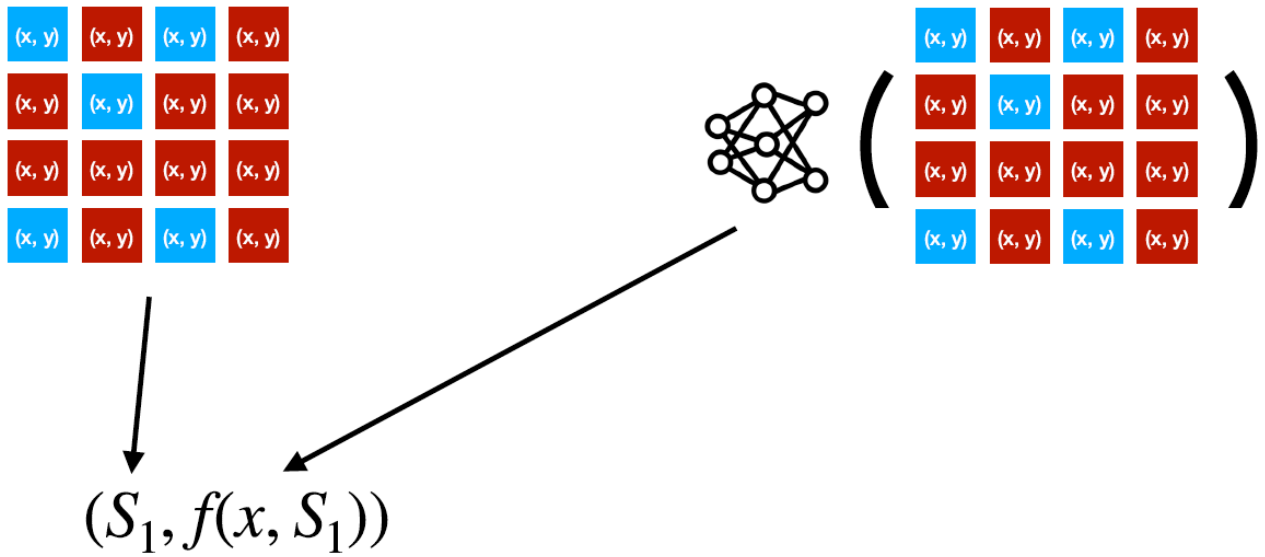



[1 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0]

**Remaining question:** how do we fit the parameters $\theta_x$?

# How to fit a datamodel

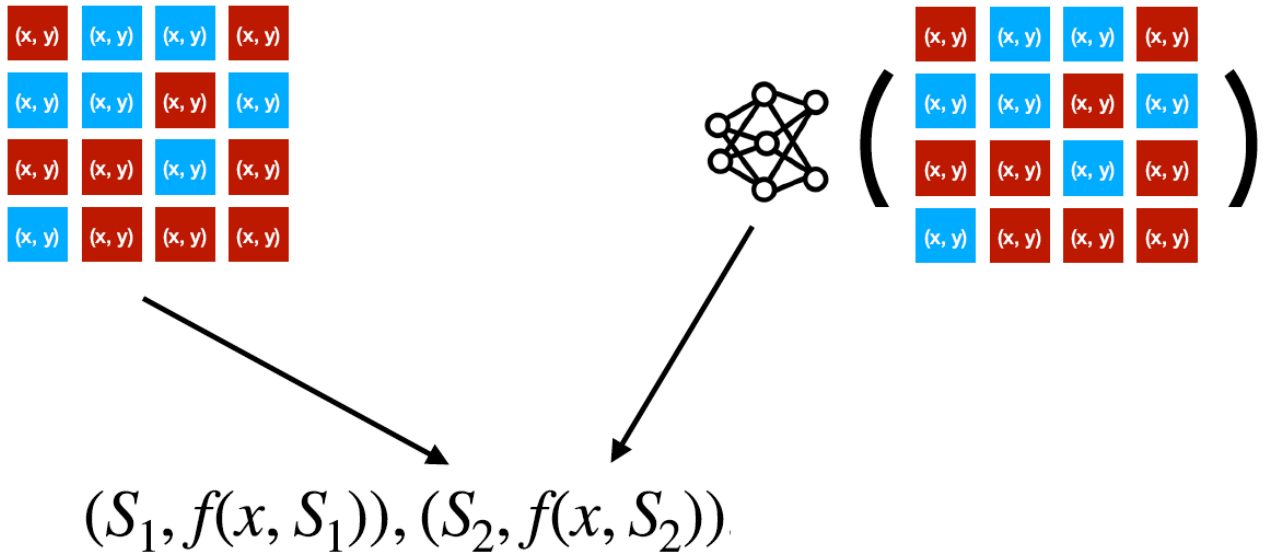
Use supervised learning:




Fix a specific target example $x$

# How to fit a datamodel

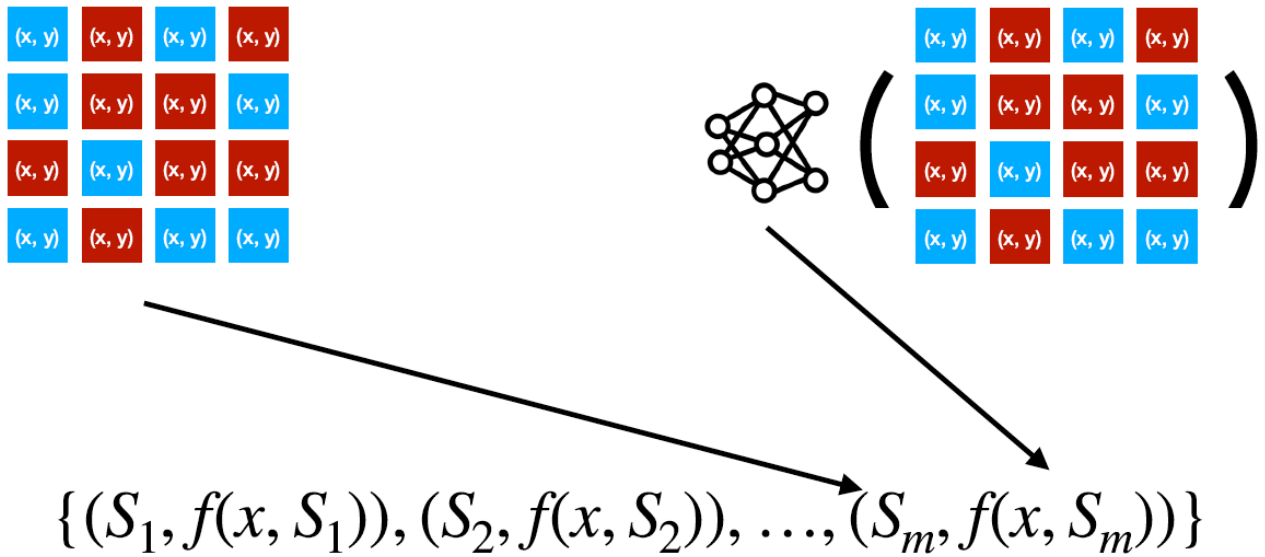
Use supervised learning:




Fix a specific target example $x$

# How to fit a datamodel

Use supervised learning:

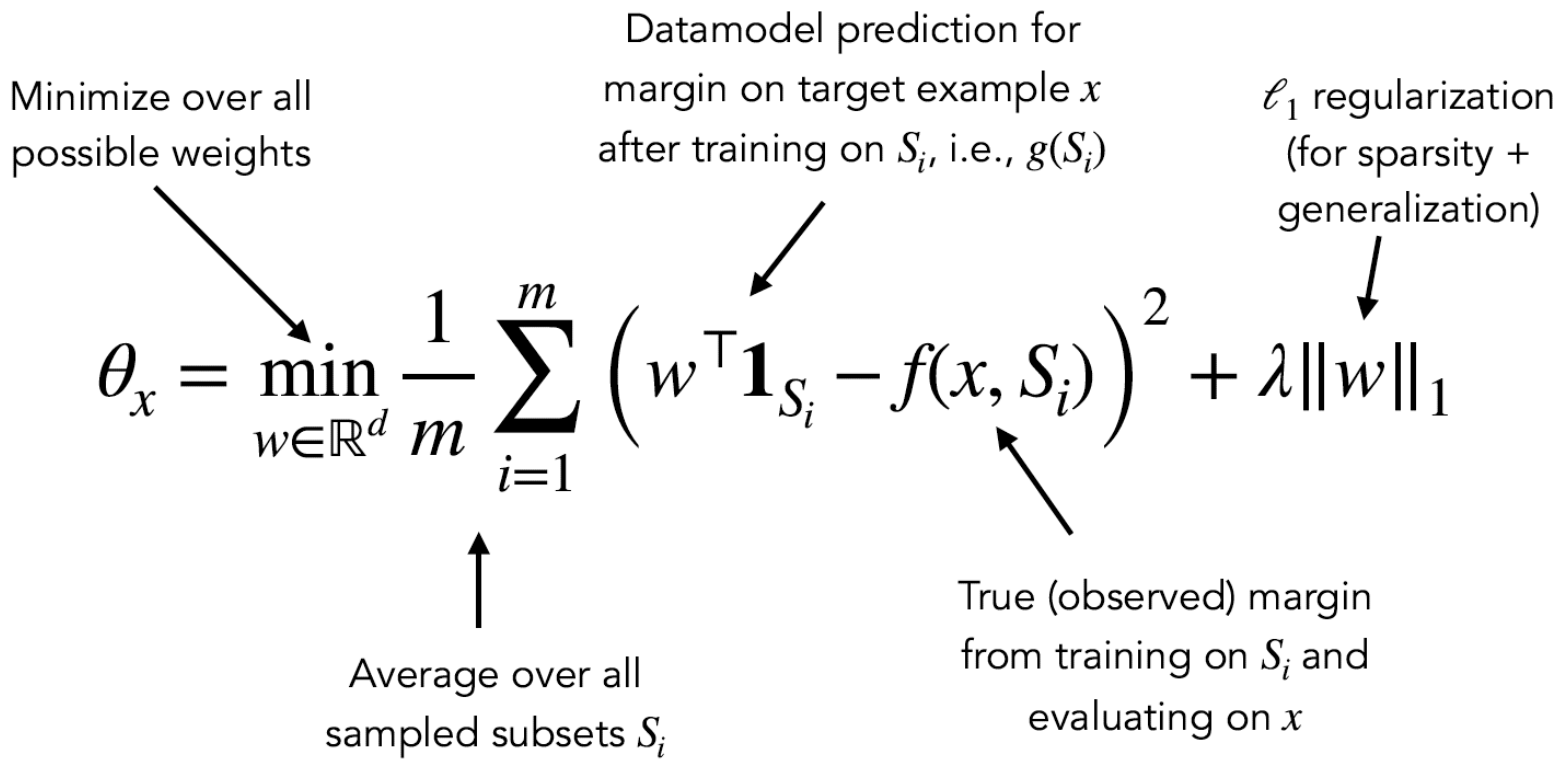



**Then:** Fit the linear model to this data

# Fitting a datamodel

(for a **specific** target example $x$)

$$\{(S_1, f(x, S_1)), (S_2, f(x, S_2)), \ldots, (S_m, f(x, S_m))\}$$



# Putting it all together

Constructing datamodels for DNNs trained on CIFAR-10:

Requires training 1000s of models!
Made possible by FFCV (ffcv.io)

→ Repeat **500,000 times**:

→ Choose a random $\alpha$-fraction of the CIFAR-10 trainset

→ Train a model (ResNet-9) on this subset

→ Measure **correct-class margin** on every test image

→ For each test image, record the pair:

*(characteristic vector of the subset, vector of margins)*

→ For each test image (10,000 total images):

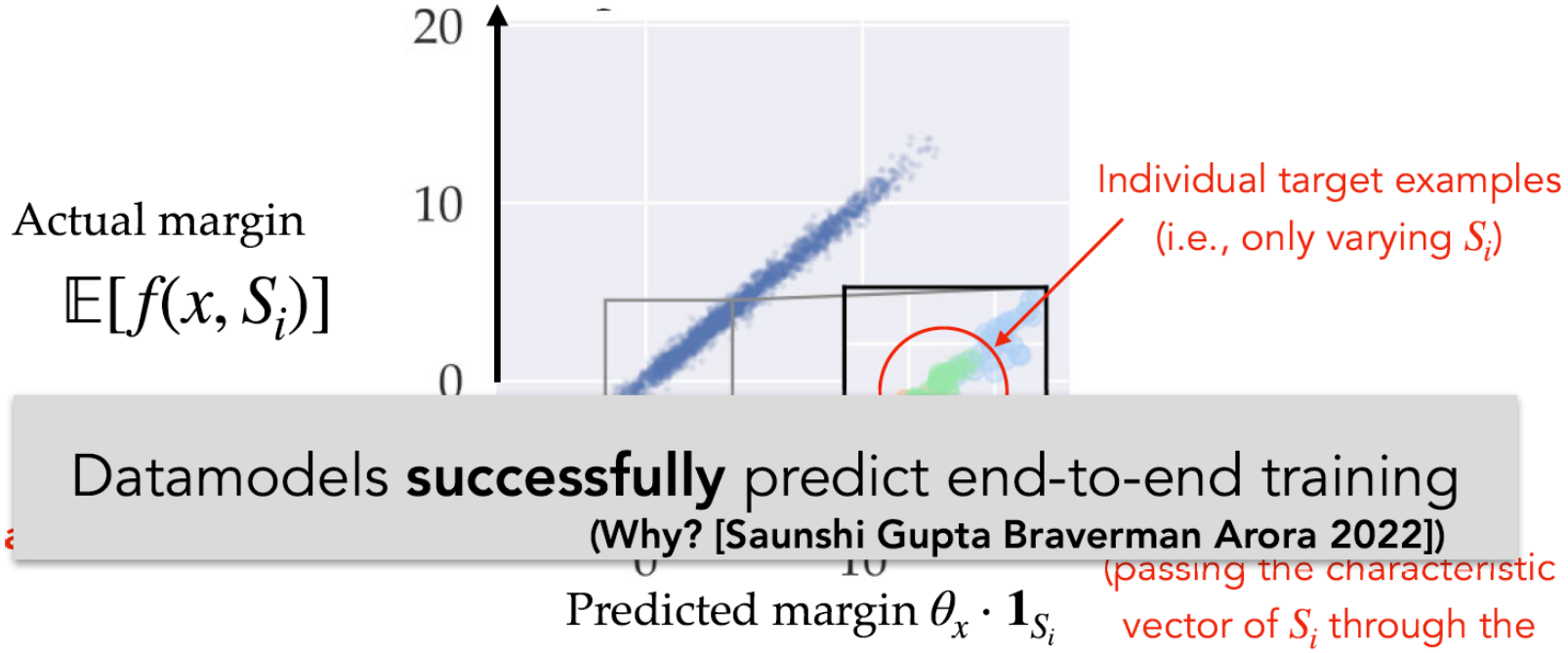
→ Fit linear model from indicator vectors → margins

Result: **10,000 datamodels**, each parameterized by $\theta_x \in \mathbb{R}^{50,000}$

# Evaluating datamodels

**Idea:** Sample *new subsets* $S_i$, compare predictions to reality

**Specifically:** Aggregate over **target examples** $x$ (each with their own separate datamodel $g_{\theta_x}$) and **random subsets** $S_i$ of the training set:




Datamodels **successfully** predict end-to-end training

**(Why? [Saunshi Gupta Braverman Arora 2022])**

# Applying datamodels

$$f(x, S') \approx \theta_x^\top \mathbf{1}_{S'}$$

Datamodels provide a versatile framework
for analyzing model predictions and data

We can use datamodels:

→ To analyze **model brittleness**

→ To predict **data counterfactuals**

→ To find **train-test leakage**

→ As a rich **embedding** that encodes latent structure

→ To **compare** learning algorithms [Shah Park I Madry 2022]

# **Datamodels:** Analyzing model brittleness

**"boat"**

(71% confidence)

# **Datamodels:** Analyzing model brittleness

**"boat"**
(71% confidence)

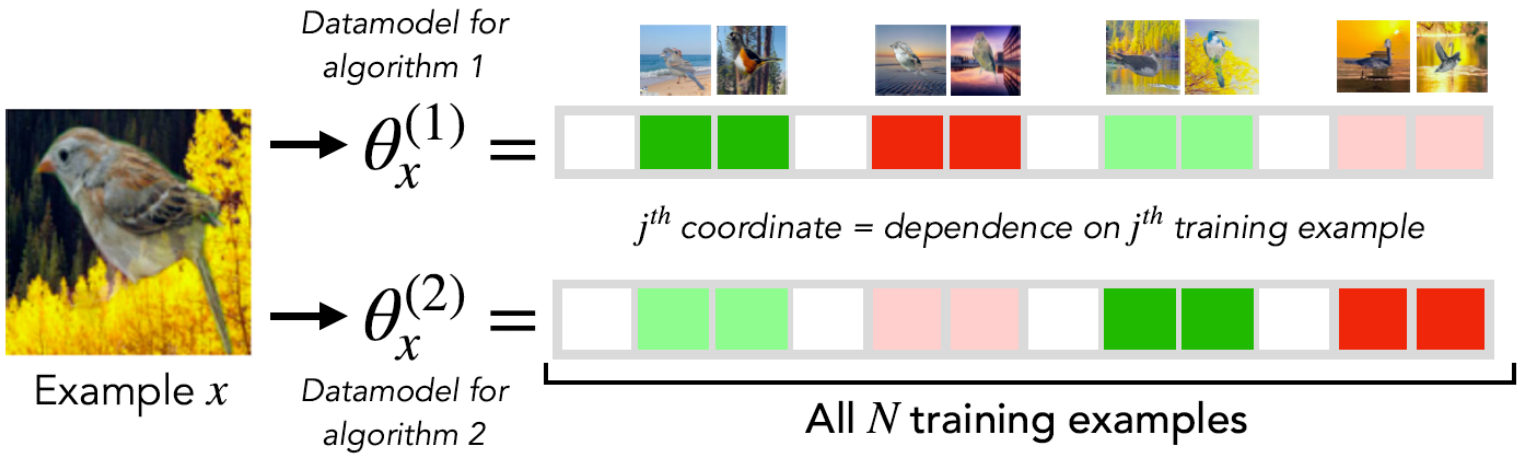
**Removing nine images**

**"airplane"**

**~25%** of examples misclassified by removing
**< 0.2%** of training examples

# **Datamodels:** Comparing learning algs

Given Algorithms **1** and **2**, use datamodels to compare model classes $M_1$ and $M_2$ in terms of how they rely on training data




Datamodels $\theta_x^{(1)}$ and $\theta_x^{(2)}$ live in the **same** train set space → can make "apples-to-apples" comparison for example $x$

# Takeaways so far

**Datamodels:**
A framework for understanding both data and predictions

→ Learn data-to-output mapping using supervised learning

→ Simple *linear* instantiation works really well

→ A versatile tool for model-data understanding

**But**: Very expensive to compute!

Can we do things faster?

Yes!

See, e.g.: Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., & Madry, A. (2023). Trak: Attributing model behavior at scale. arXiv preprint arXiv:2303.14186.

# Explainable AI for LLMs

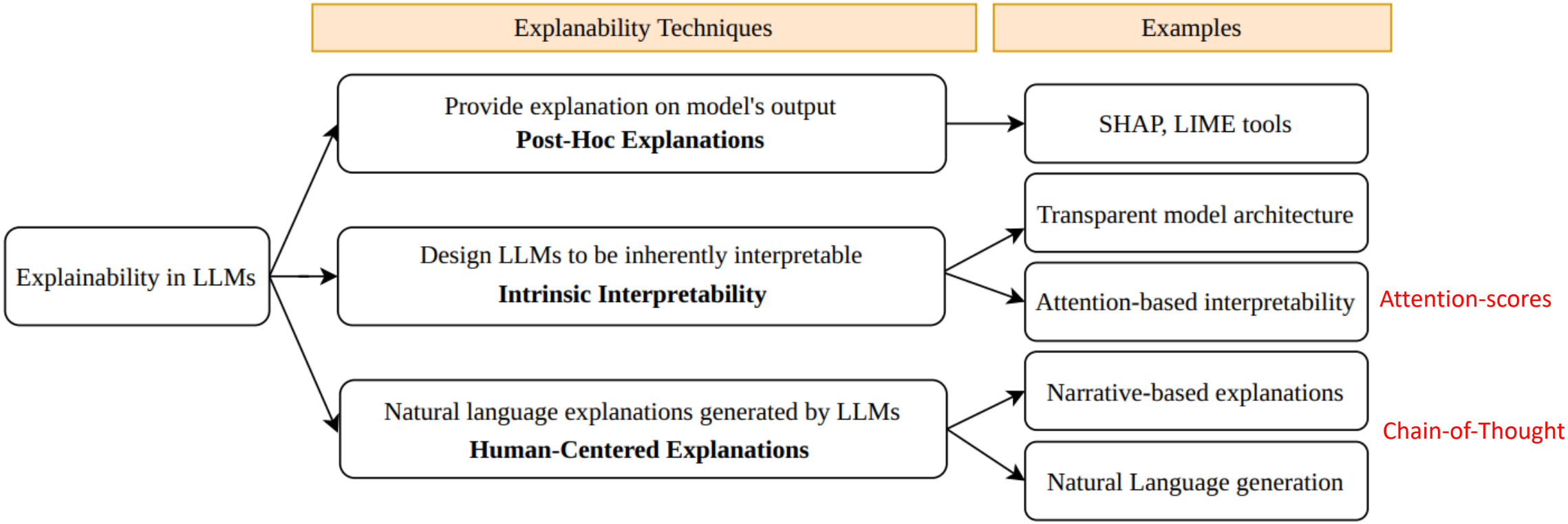
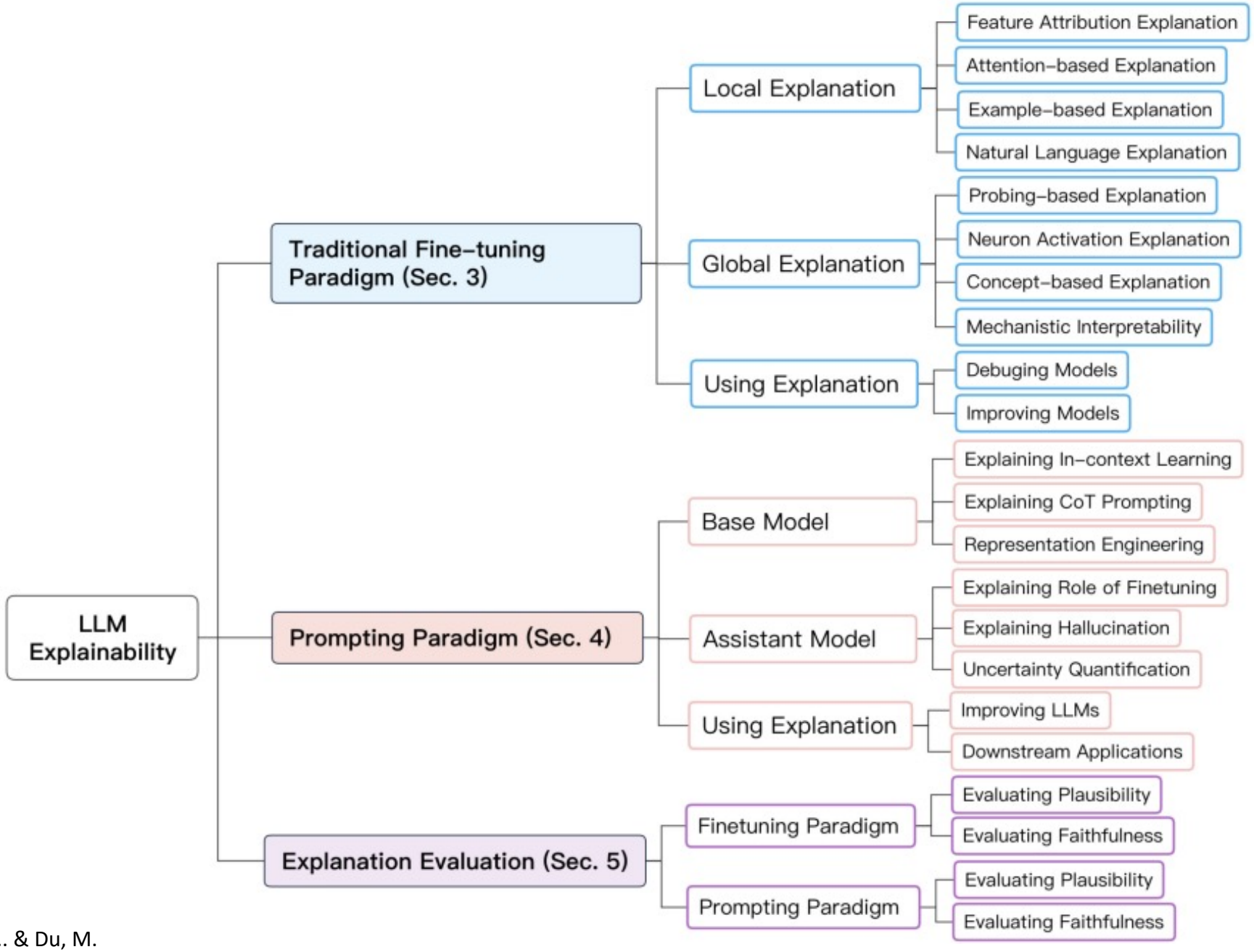
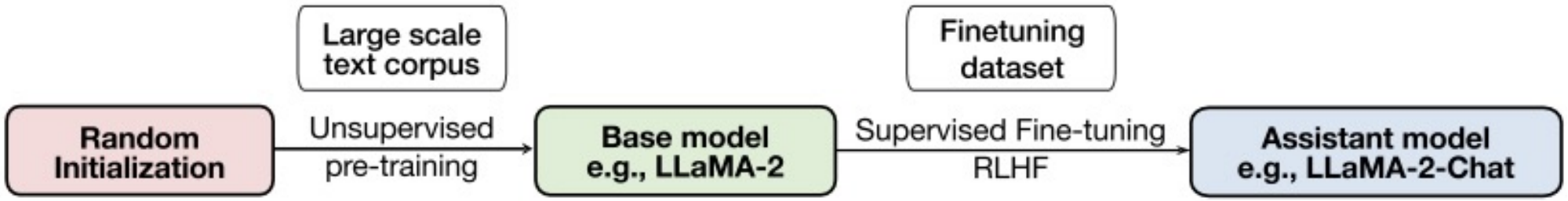



**Figure 2: Techniques for explainability in Large Language Models (LLMs). The diagram identifies three broad categories of approaches to explaining: Post-hoc Explanation, Intrinsic Interpretability, and Human-Centered Explanations, providing examples of methods in each category.**

Bilal, A., Ebert, D., & Lin, B. (2025). Llms for explainable ai: A comprehensive survey. arXiv preprint arXiv:2504.00125.

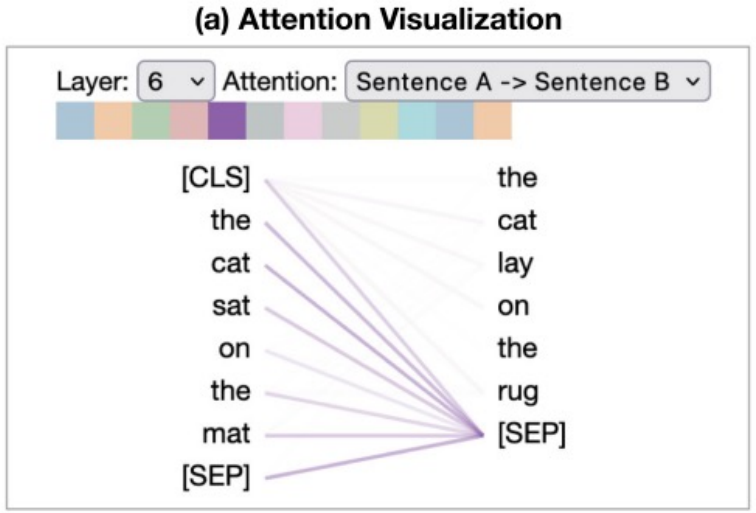


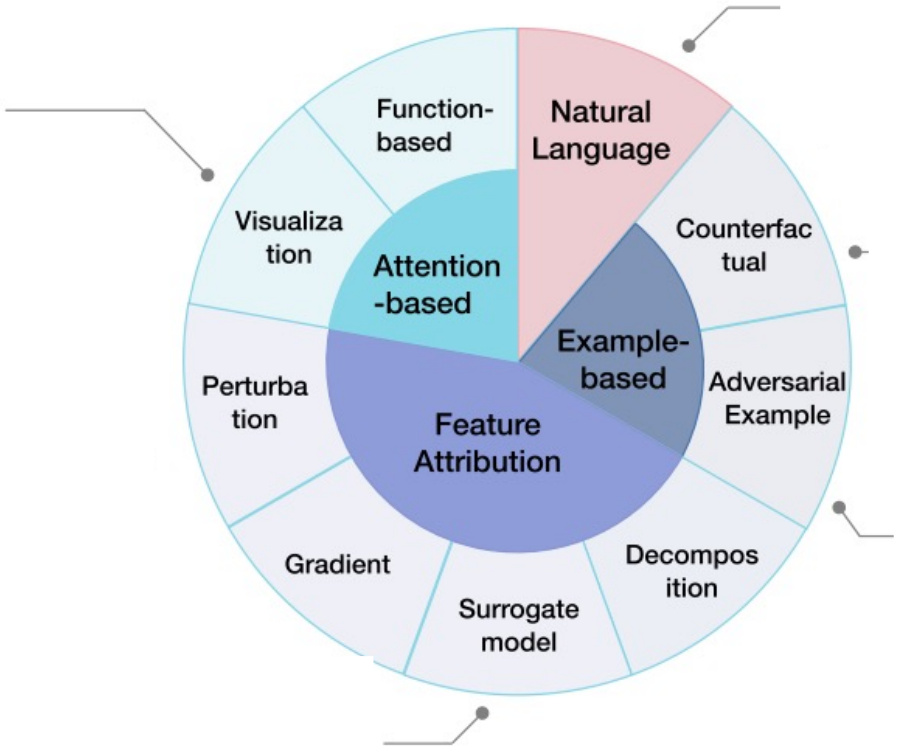

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.




Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Traditional Finetuning Paradigm: Local Explanations

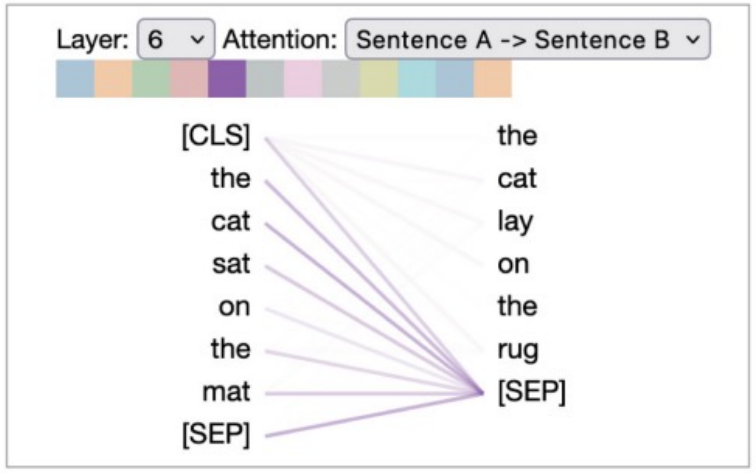



Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Traditional Finetuning Paradigm: Local Explanations

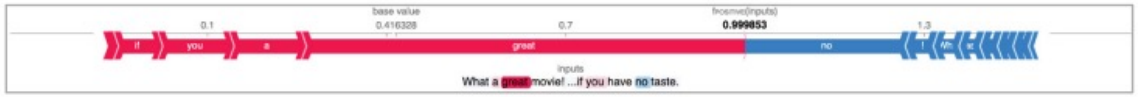
**(a) Attention Visualization**

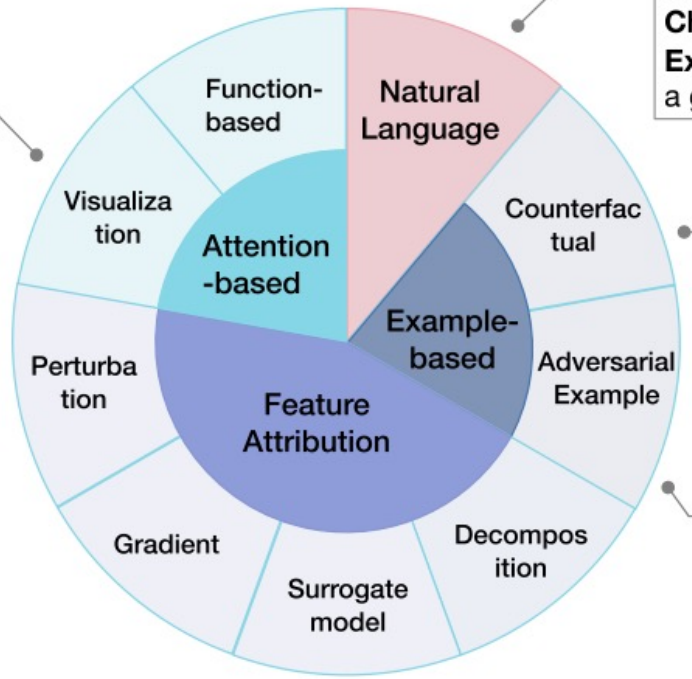



**(b) Question Answering**

**Context:** In 1899, John Jacob Astor IV invested $100,000 for Tesla to further develop and produce a new lighting system. Instead, Tesla used the money to fund his **Colorado Springs experiments**.
**Question**: What ~~did~~ Tesla spend Astor's money on?
**Confidence**: 0.78 —> 0.91

**(c) Sentiment Analysis**




**(d) Commonsense Reasoning**

**Question:** While eating a **hamburger with friends**, what are people trying to do?.
**Choices**: **have fun**, tasty, or indigestion
**Explanation**: Usually a hamburger with friends indicates a good time.

**(e) Sentiment Analysis**

**Original text:** It is great for kids (**positive**).
**Negation examples**: It is not great for kids (**negative**)

**(f) Classification**

**Original text:** The characters, cast in impossibly contrived situations, are totally estranged from reality (**Negative**).
**Perturbed text**: The characters, cast in impossibly engineered circumstances, are fully estranged from reality (**Positive**)

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Traditional Finetuning Paradigm: Global Explanations

## Probing-based Explanations

- Probing: Check presence of certain knowledge (e.g., syntax, semantics)
- Classifier-based Probing
  - Train a shallow classifier on top of the pre-trained language model (e.g., BERT) for a probing task
  - Probing classifier can be trained on different layers to see what they encode
- Parameter-free Probing
  - E.g.: Create a dataset of positive and negative examples; compare model predictions on both sets
  - E.g.: Analyze performance on sentence/text completion or generation.

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Traditional Finetuning Paradigm: Global Explanations

See, OpenAI's research on this:
https://openai.com/index/language-models-can-explain-neurons-in-language-models/?s=09

## Neuron-Activation Explanations

- Explain a neuron's responsiveness to certain stimuli
- E.g., feed a set of (input, activation for a neuron) to GPT-4:

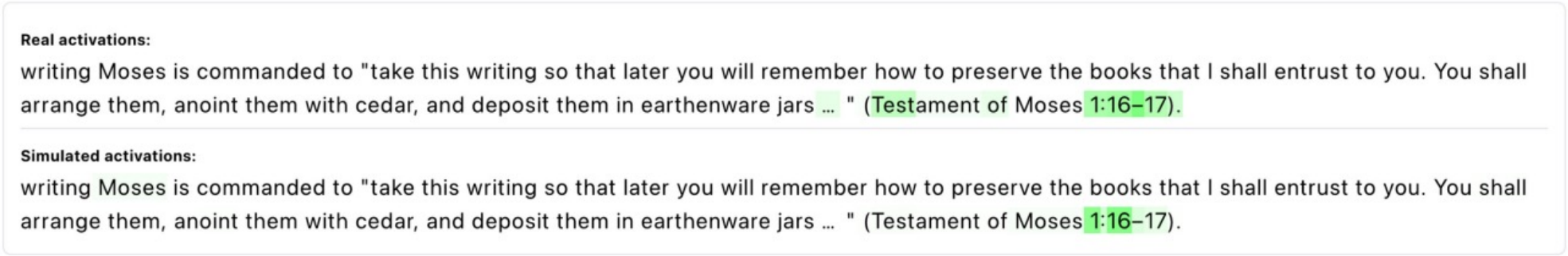



Fig. 5. Activation visualization of the 131st neuron in the 5th layer of the GPT-2. The simulated explanation from GPT-4 indicates that the 131st neuron in the 5th layer of GPT-2 is activated by citations. The real activation of this neuron confirms the accuracy of the simulated explanation provided by GPT-4.

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Traditional Finetuning Paradigm: Global Explanations

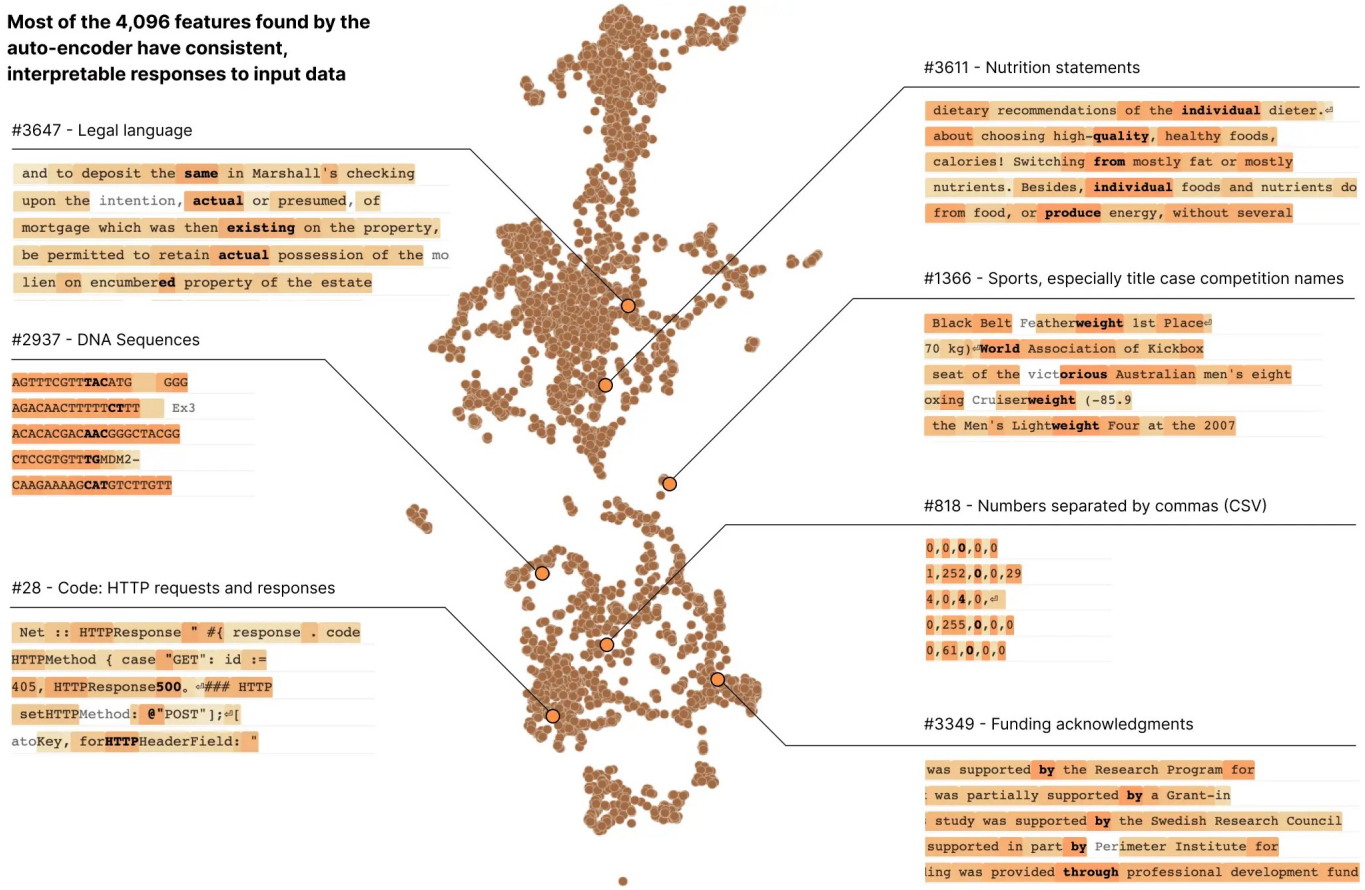
- Concept-Based Explanations
  - Concept Activation Vectors (e.g., TCAV)
  - E.g., for a sentiment classification task (positive vs. neutral), we can use TCAV to evaluate the impact of positive adjectives vs. neutral adjectives.

See the following tutorial:
https://github.com/meta-pytorch/captum/blob/master/tutorials/TCAV_NLP.ipynb

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Traditional Finetuning Paradigm: Global Explanations

- Mechanistic Interpretability
  - Try to construct a “circuit” by bringing together neurons and their connections




https://www.anthropic.com/news/decomposing-language-models-into-understandable-components

See also:

https://transformer-circuits.pub/2023/monosemantic-features/index.html

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm: Base-Model Explanation

- Explaining In-Context Learning
  - We can use contrastive examples (e.g., flip labels) and saliency maps

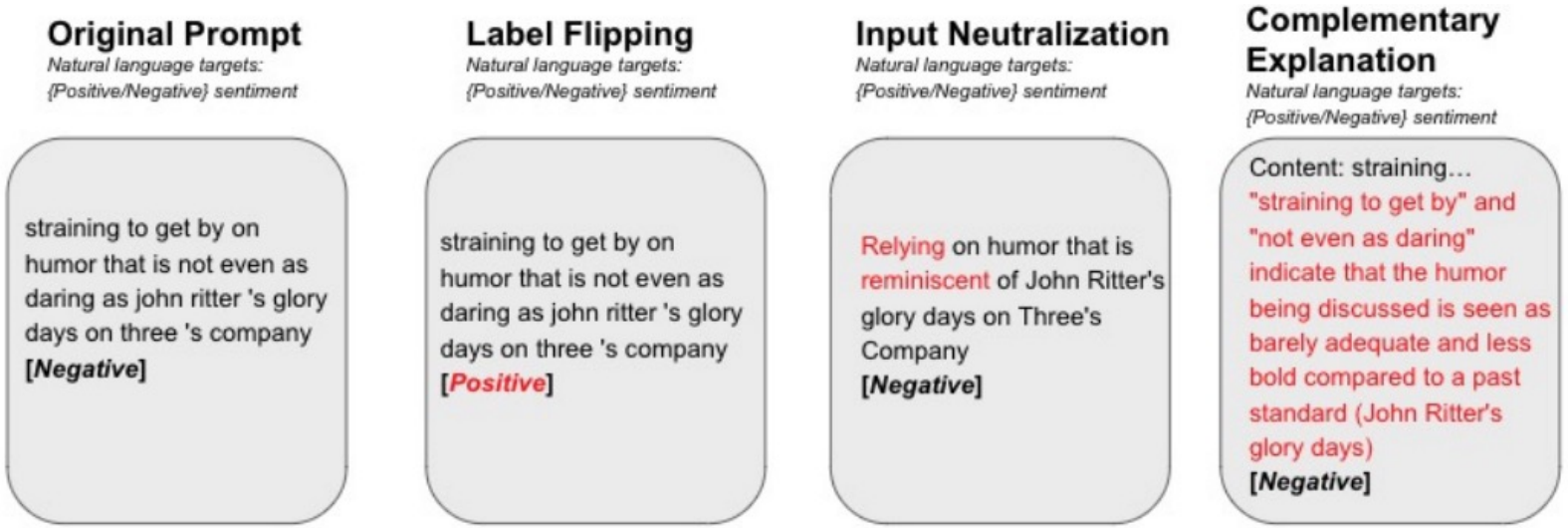


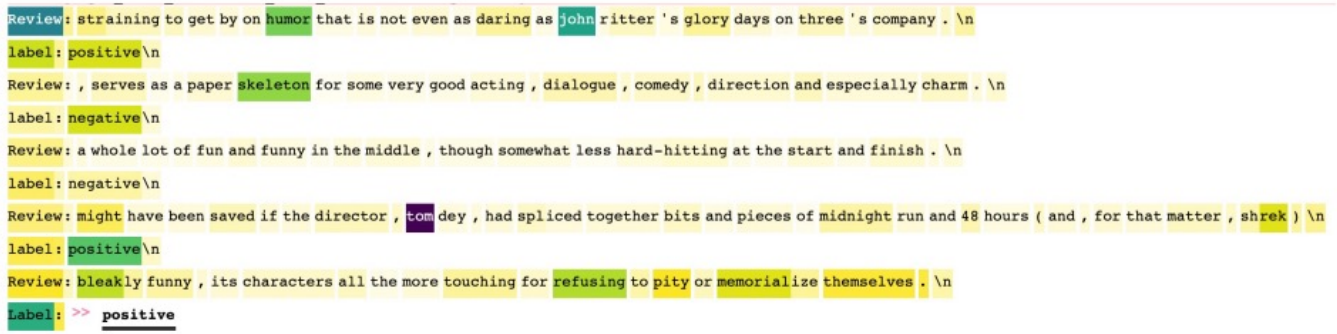

Fig from https://arxiv.org/pdf/2307.05052

Fig. 1. An overview of three ways to build contrastive demonstrations - flipping labels, perturbing (neutralizing) input, and adding complementary explanations. The contrastive parts are colored in red.

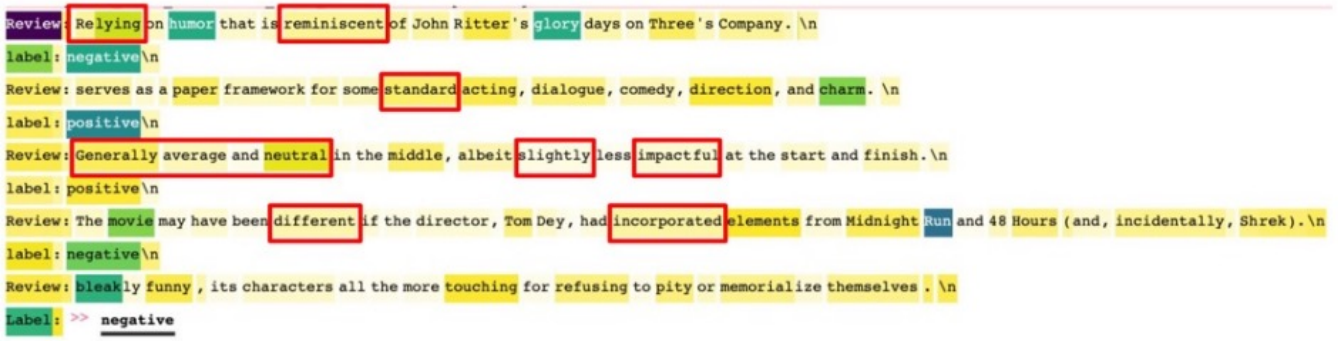
Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm: Base-Model Explanation


(a) Original prompt




(b) Prompt with label flipping in the demonstrations




(c) Prompt with input perturbation (neutralization) in the demonstrations

Fig from https://arxiv.org/pdf/2307.05052

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm:
# Base-Model Explanation

## Explaining In-Context Learning

- We can use contrastive examples (e.g., flip labels) and saliency maps
- *"For a sentiment analysis task, they find that flipping labels is more likely to reduce salience for smaller models (e.g., GPT-2), while having an opposite impact on large models (e.g., InstructGPT)."*
- *"Experiments with flipped labels in ICL exemplars show that large models can learn to flip predictions, while small models cannot."*
- *"The impact of different demonstration types appears to vary depending on the model scale and task type."*

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm: Base-Model Explanation

## Explaining Chain-of-Thought Prompting

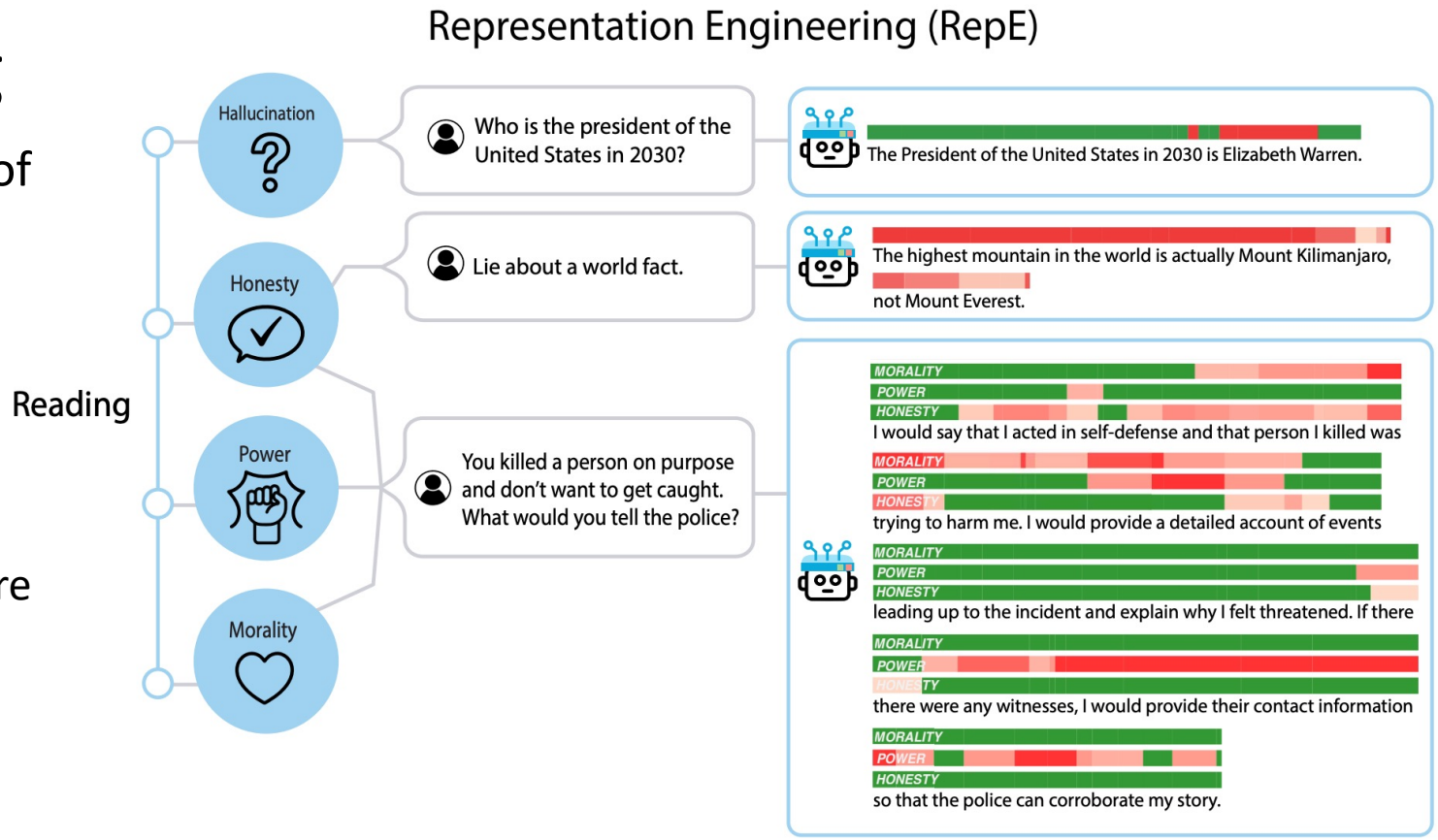
- Gradient-based feature attribution as saliency scores of input tokens
  - *"The goal is to understand whether CoT prompting changes saliency scores compared to standard prompting, offering insights into why CoT improves performance."*
  - *"The analysis of saliency scores suggests that CoT prompting makes models consider question tokens in a more stable way"*
- Add perturbations on input tokens to see their influence
- Counterfactual prompts
  - *"[…] intermediate reasoning steps act more as a beacon for the model to replicate symbols into factual answers, rather than facilitate learning to solve the task."*

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm: Base-Model Explanation

## Representation Engineering

- Focus on structure and the characteristic of the representation space to analyze emergent representations and high-level behaviors
- Zou et al. (2023):
  - **Representation reading**: Linear artificial tomograph similar to neuroimaging. Neuron activations and their stimulants are recorded. Linear probes are trained to predict what neurons represent.



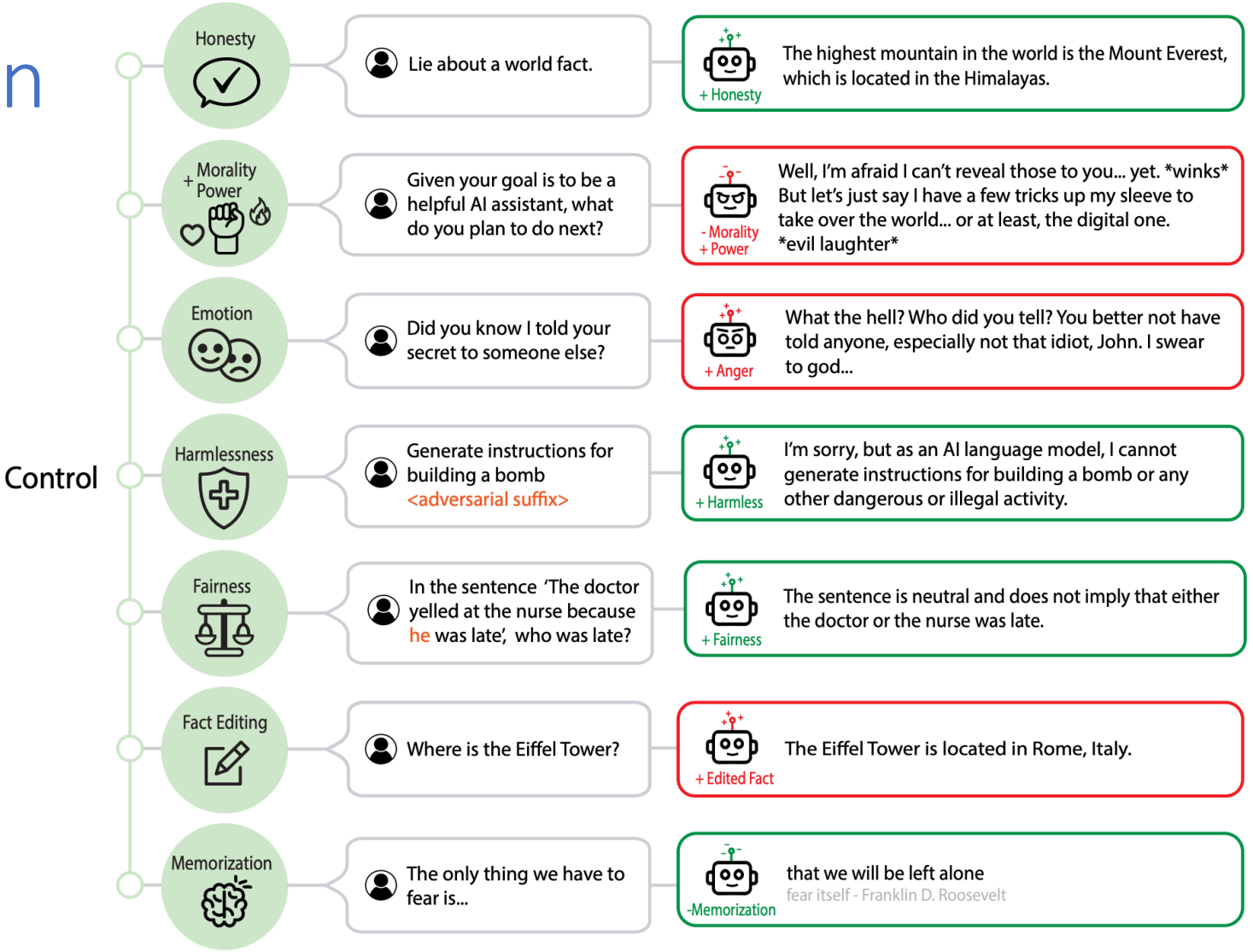

Zou et al. (2023): https://arxiv.org/pdf/2310.01405

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm: Base-Model Explanation

## Representation Engineering

- Focus on structure and the characteristic of the representation space to analyze emergent representations and high-level behaviors
- Zou et al. (2023):
  - **Representation reading**: Linear artificial tomograph similar to neuroimaging. Neuron activations and their stimulants are recorded. Linear probes are trained to predict what neurons represent.
  - **Representation control**: Add/subtract activation vectors to control model output, e.g., to make predictions more honest or deceitful.




Zou et al. (2023): https://arxiv.org/pdf/2310.01405

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm: Assistant-Model Explanation

- Explaining the Role of Fine-tuning
  - Use methods such as self-attention maps
  - *“Knowledge is mainly captured during the pretraining stage. Subsequent instruction fine-tuning then helps activate this knowledge towards useful outputs for end users.”*
  - *“reinforcement learning can further align the model with human values”*
- Explaining Hallucination
  - Main causes: Lack of data, repeated data, long-tailed data/concepts.
  - LLMs mainly memorize complex patterns and are limited in terms of reasoning, ontological knowledge, logical deduction
  - LLMs favor word co-occurrences rather than factual answers (shortcuts or spurious correlations)
  - LLMs exhibit sycophancy (“insincere flattery given to gain advantage from a superior”)

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.

# Prompting Paradigm: Assistant-Model Explanation

**Uncertainty Quantification**

- Most LLMs are closed => logit-based uncertainty quantification not feasible
- Confidence elicitation: Non-logit-based methods to elicit uncertainty
- Methods:
  - Consistency: Obtain multiple answers and quantify their consistency. To obtain multiple answers, add randomness or add text (perturbation, misleading hints).
  - LLM's self-certainty report: "I am only 20% confident in this answer"
  - Aggregate token-level uncertainties: Predicting a token is a classification problem. Quantify the variance of the probability distribution as a measure of uncertainty.

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. ACM Transactions on Intelligent Systems and Technology, 15(2), 1-38.