Lecture 21: Explainability

Trustworthy Machine Learning

Spring 2024

Explainability

- Recap: Feature Attribution Methods
  - LIME (Local Interpretable Model-agnostic Explanations) algorithm
  - SHAP methods based on cooperative game theory
  - Saliency Maps (different versions)
  - Formal guarantees for feature attribution methods

- Today's agenda:
  - Counterfactuals
  - Representation-based explanations

- Resource: Tutorial on "Interpreting ML models" by Hima Lakkaraju

Counterfactual

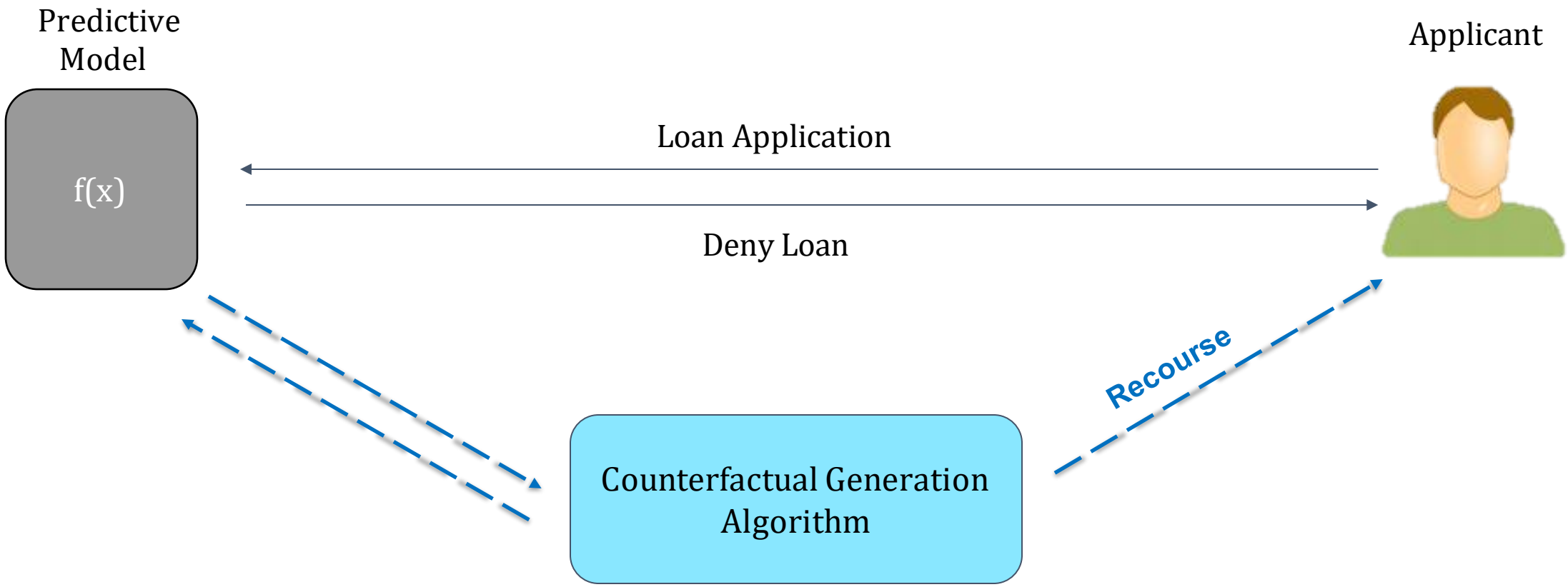
- As ML models are increasingly deployed to make high-stakes decisions (e.g., loan applications), it becomes important to provide recourse to affected individuals.

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).

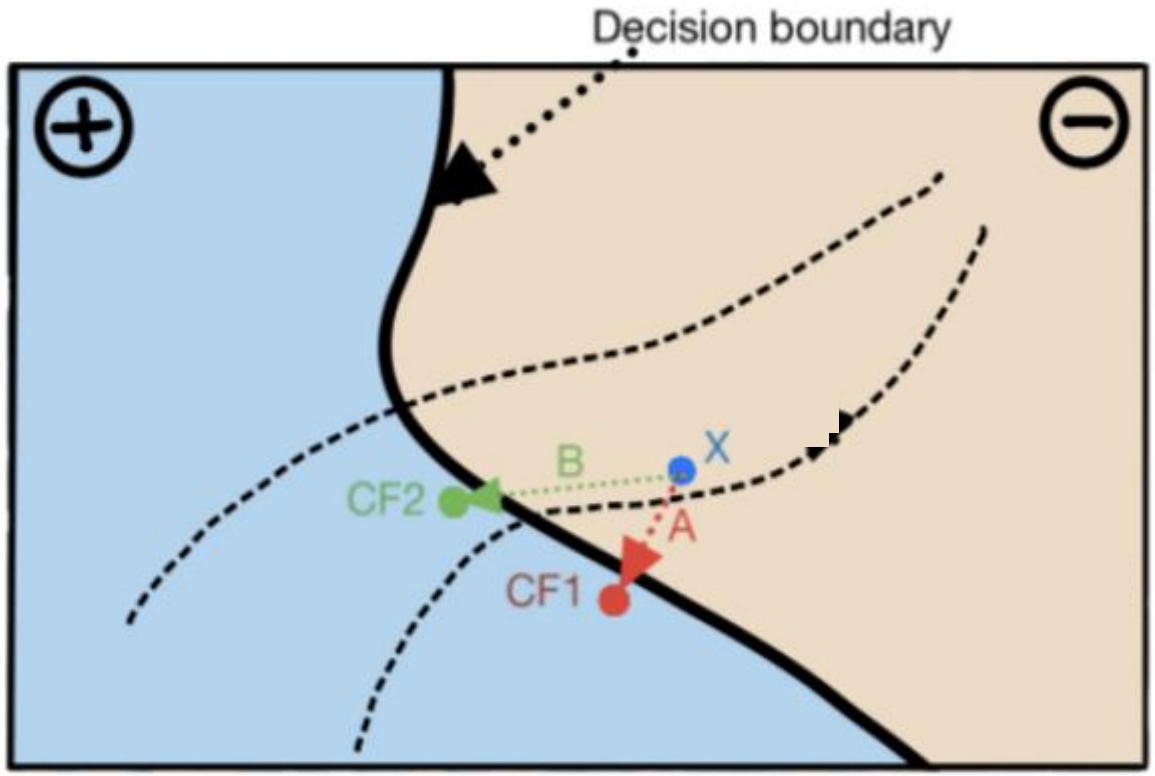
- Key publications:
  - Counterfactual visual explanations; Goyal et al.; ICML 2019
  - Counterfactual explanations without opening the black box; Wachter et al.; 2018

Counterfactual Explanation



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

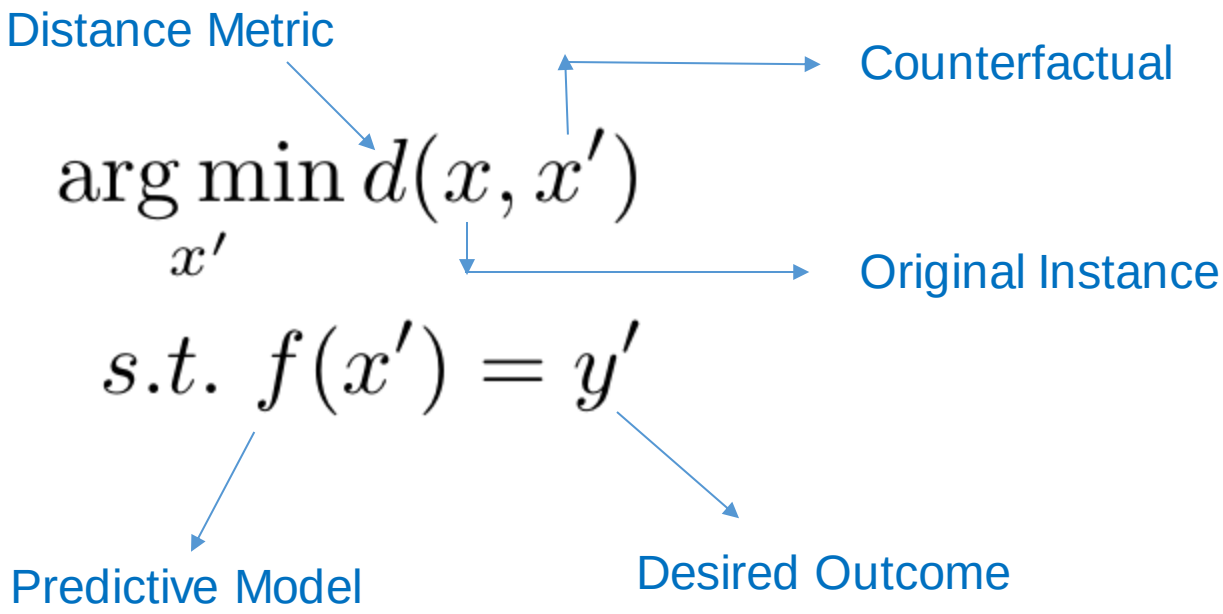
Generating Counterfactual



Proposed solutions differ on:

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

Minimum Distance Counterfactuals

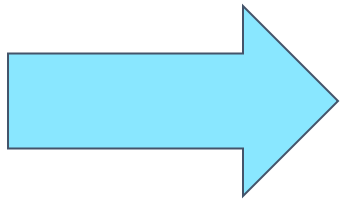


Choice of distance metric dictates what kinds of counterfactuals are chosen.

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

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.

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).

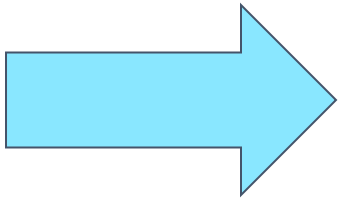
Person 4: If your LSAT was 35.8, and you were 'white', you would have an average predicted score (0).

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

Not feasible to act upon these features!

Feasible Least Cost Counterfactuals

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

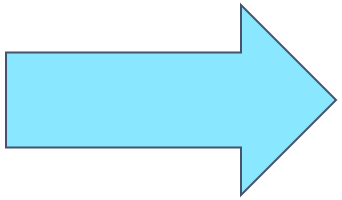


$$\arg\min_{x' \in \mathcal{A}} cost(x, x') \quad s.t.\ f(x') = y'$$

- $\mathcal{A}$ 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

Feasible Least Cost Counterfactuals

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



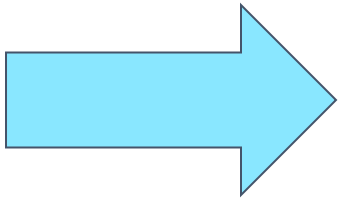
$$\arg\min_{x' \in \mathcal{A}} cost(x, x') \quad s.t.\ f(x') = y'$$

How to solve such an optimization problem ?

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

Feasible Least Cost Counterfactuals

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

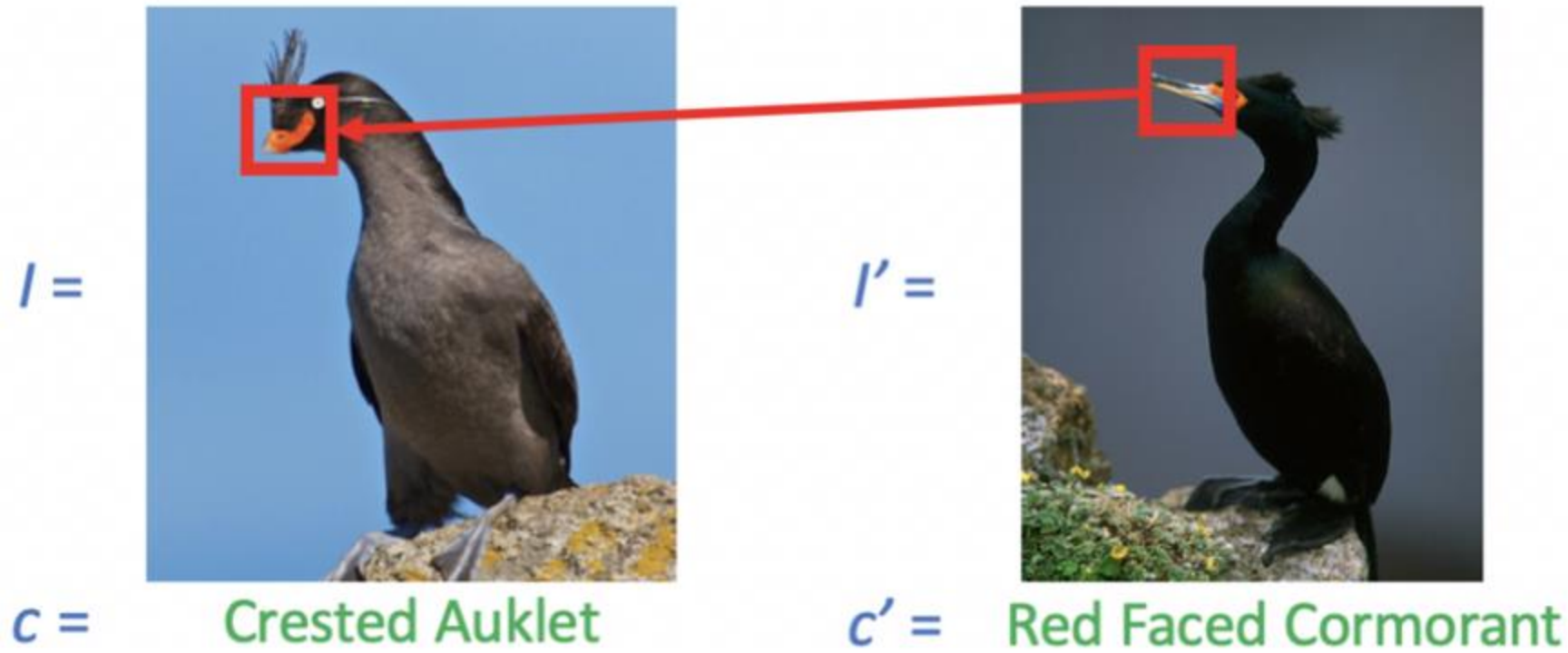


$$\arg\min_{x' \in \mathcal{A}} cost(x, x') \quad s.t.\ f(x') = y'$$

How to handle non-linear classifiers ?

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

Visual Counterfactuals



Reference: Counterfactual visual explanations; Goyal et al.; ICML 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'

Visual Counterfactual Generation problem

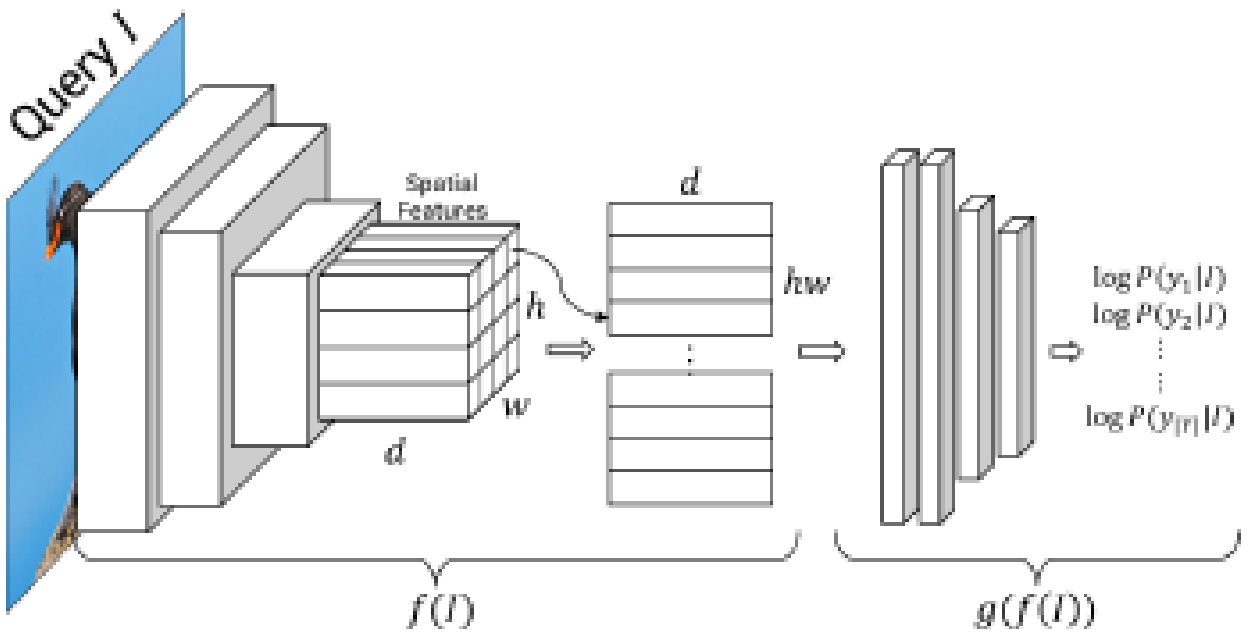


Figure 3. We decompose a CNN as a spatial feature extractor $f(I)$ and a decision network $g(f(I))$ as shown above.

Minimum-edit Counterfactual

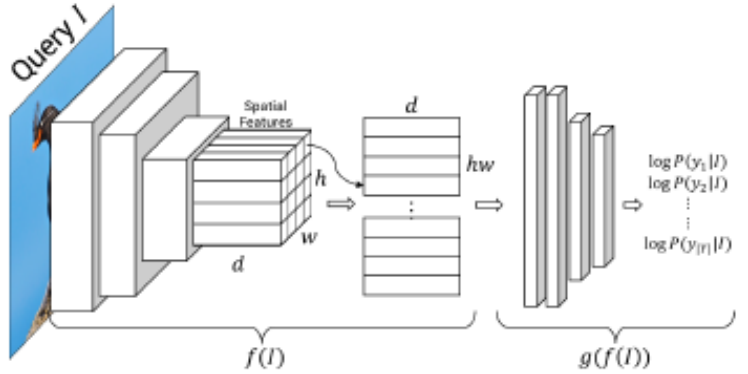


Figure 3. We decompose a CNN as a spatial feature extractor $f(I)$ and a decision network $g(f(I))$ as shown above.

- 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)

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

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

Minimum-edit Counterfactual

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

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

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

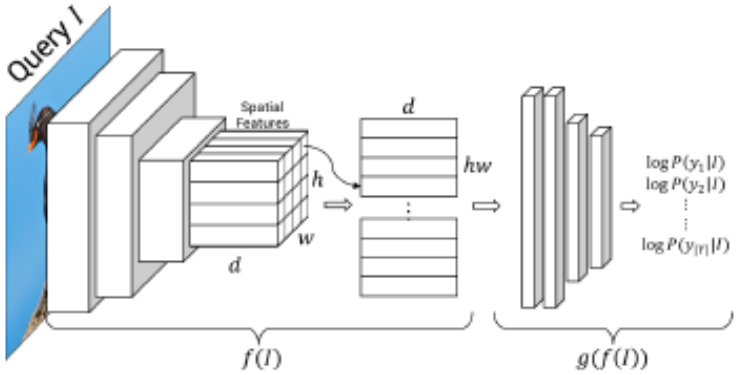


Figure 3. We decompose a CNN as a spatial feature extractor $f(I)$ and a decision network $g(f(I))$ as shown above.

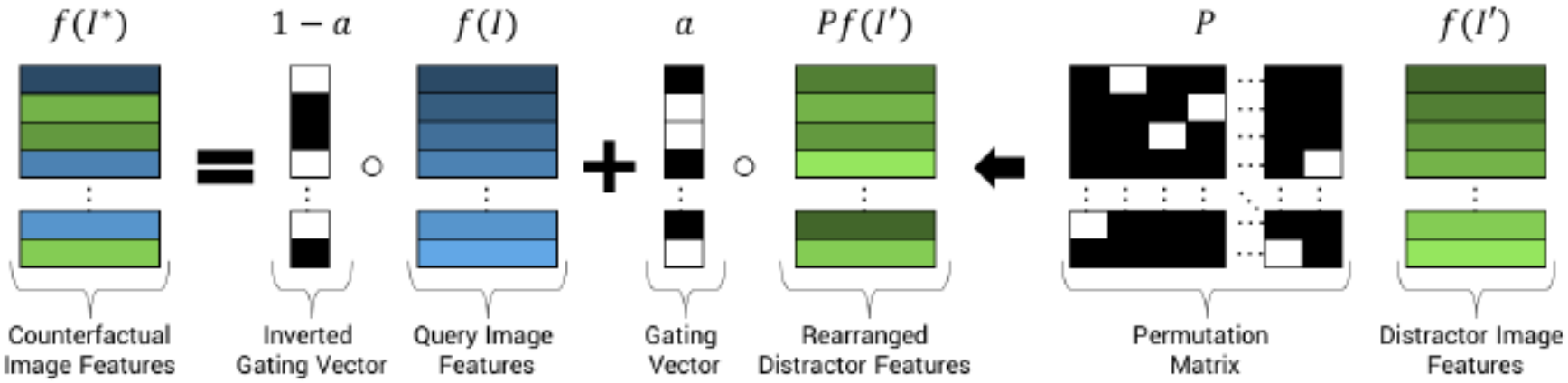


Figure 2. To parameterize our counterfactual explanations, we define a transformation that replaces regions in the query image I with those from a distractor I'. Distractor image features $f(I')$ are first rearranged with a permutation matrix P and then selectively replace entries in $f(I)$ according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

Greedy Search

$$\begin{aligned} \underset{P,\mathbf{a}}{\text{minimize}} \quad & ||a||_1 \\ \text{s.t.} \quad & c' = \text{argmax} \;\; g((\mathbb{1} - \mathbf{a}) \circ f(I) + \mathbf{a} \circ P f(I')) \\ & a_i \in \{0, 1\} \;\; \forall i \;\; \text{and} \;\; P \in \mathcal{P} \end{aligned}$$

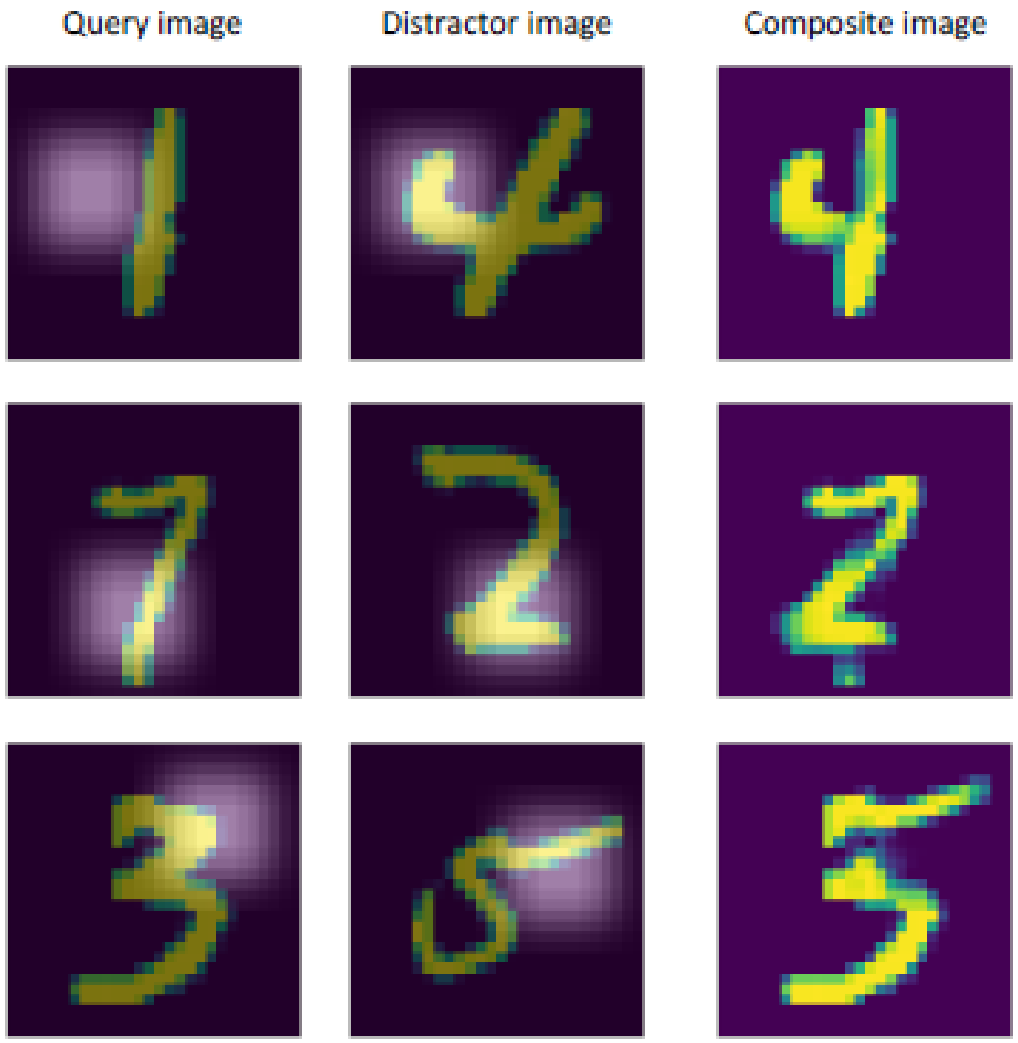
- Too many choices for permutation matrix P and gating vector a
- Greedy strategy: Do edits one at a time
- Single edit: Find one feature out of hw possibilities in f(I) to be replaced with one feature in f(I')
- Choose the edit that maximizes the prediction of class c' for the edited image

Greedy Search

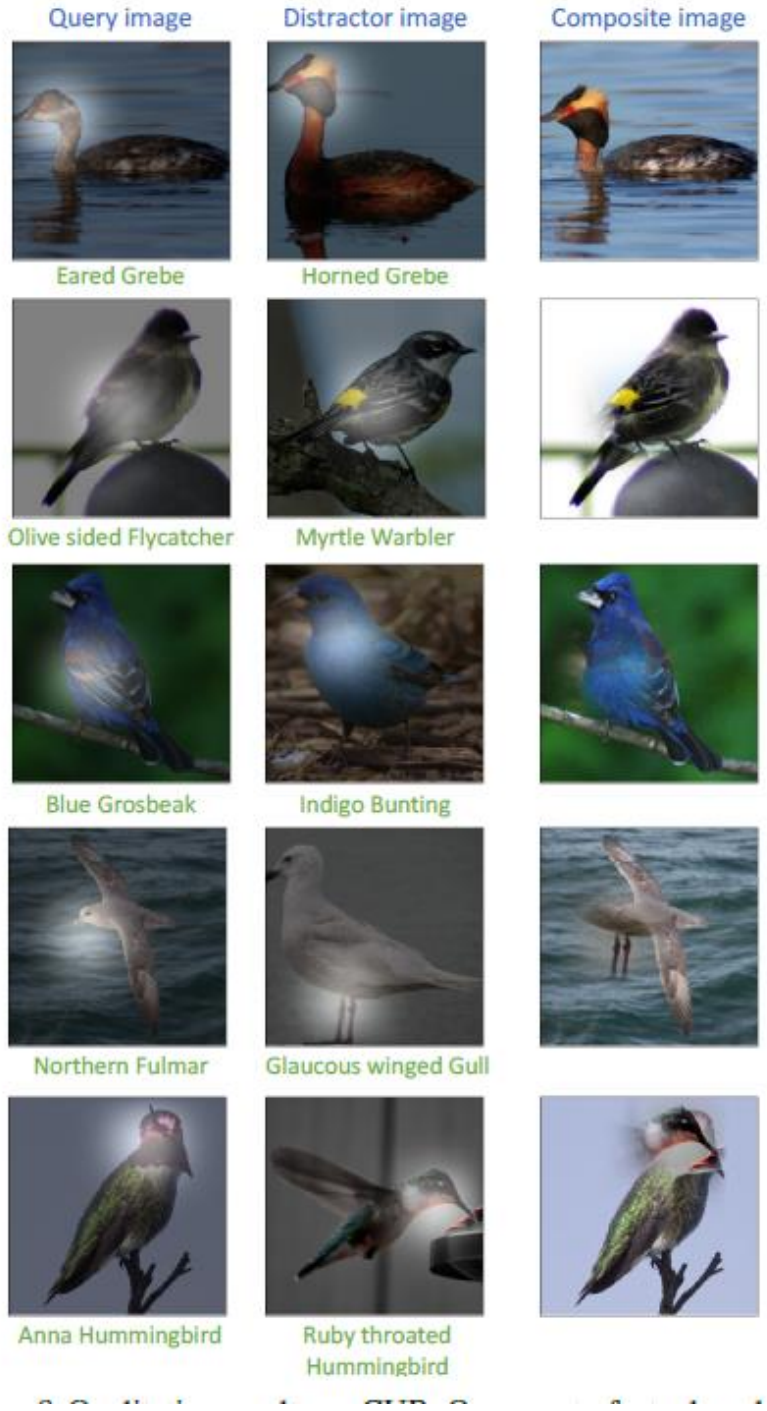
Algorithm 1 Greedy Sequential Search

Data: query image I with class c, distractor I' with class c'
Result: list of edits S that change the model decision
$S \leftarrow [\,]$ $F^* \leftarrow f(I)$ $F' \leftarrow f(I')$
/* Until decision is changed to c' */
while $c' \neq argmax\ g(F^*)$ **do**
  /* Find single best edit excluding previously edited cells in S */
  $i, j' \leftarrow \text{BestEdit}(F^*, F', S)$
  /* Apply the edit and record it */
  $F^*_{i,*} = F'_{j',*}$
  $S.append(\{i, j'\})$
end

Evaluation of CNN on MNIST Dataset



Evaluation of VGG-16 on Caltech-UCSD Birds Dataset



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

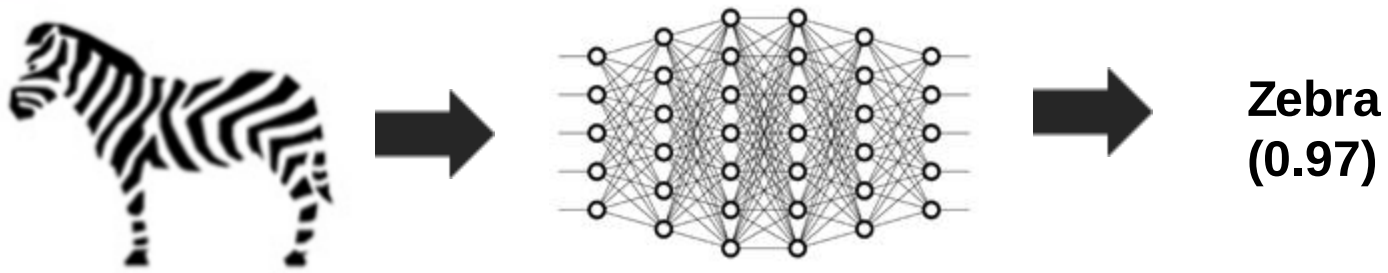
Representation-based Explanations

- 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

Concept-based Explanations

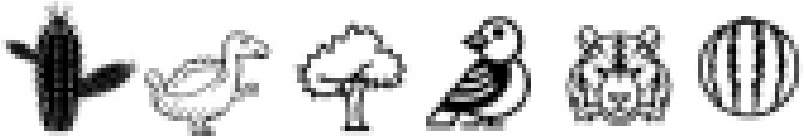


How important is the notion of “stripes” for this prediction ?

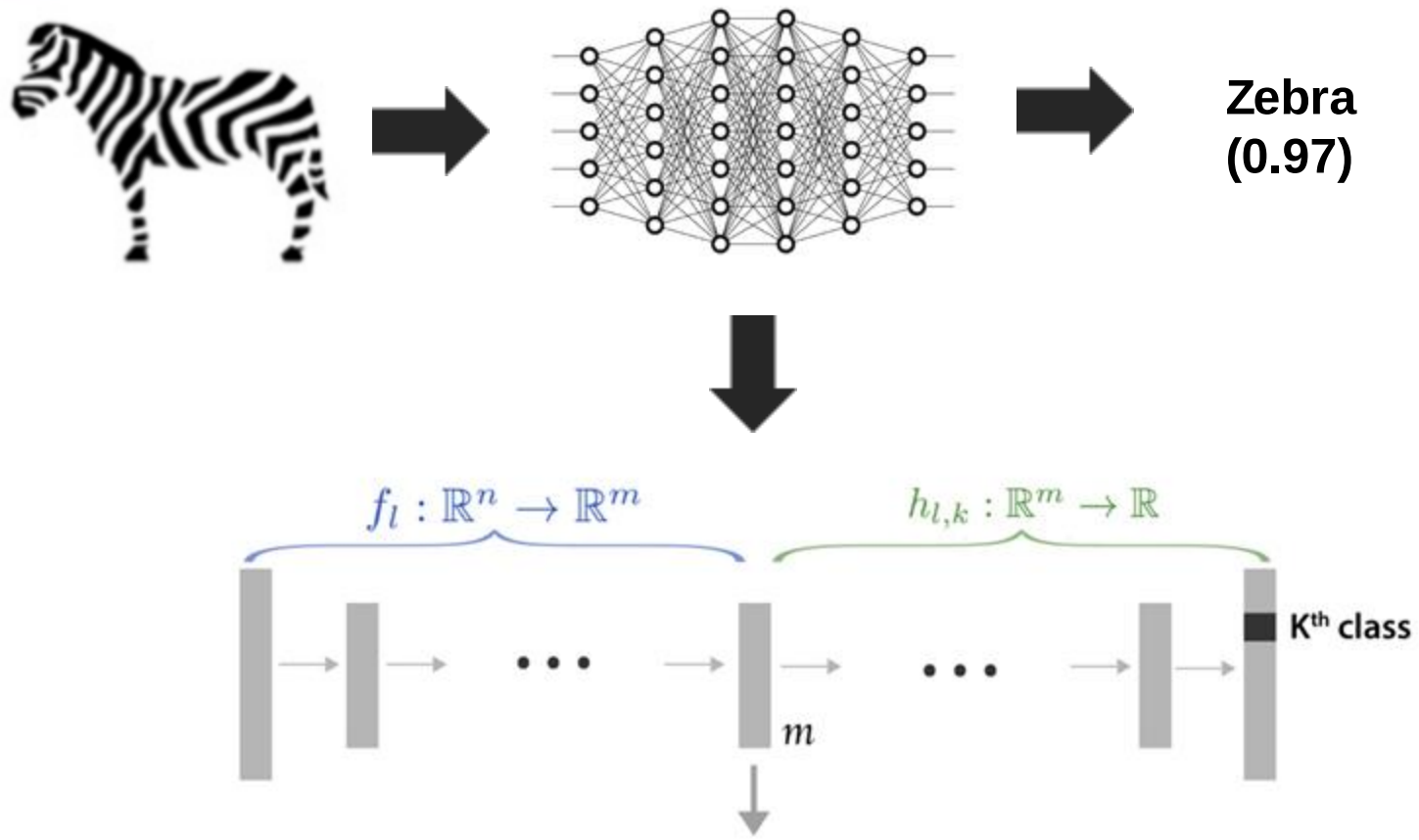
Step 1: Specifying Concepts

- 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



Internal Layers in DNN

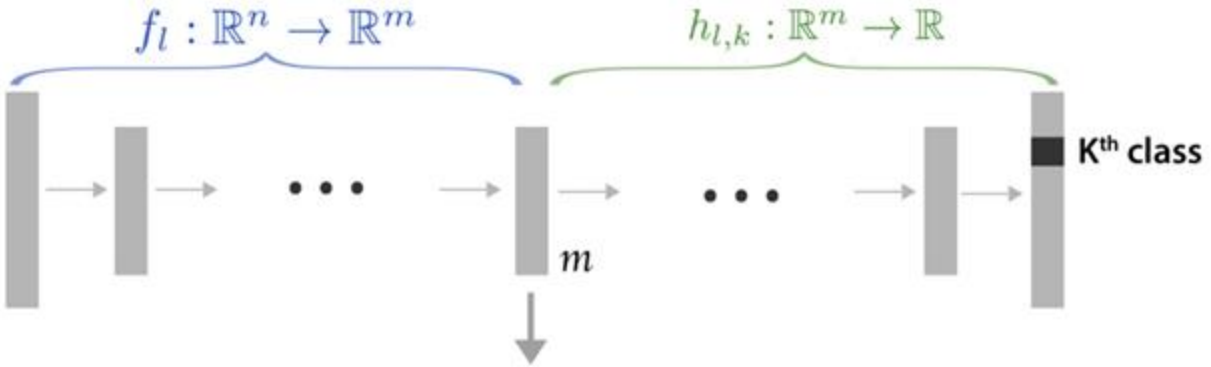


Input representation at layer l

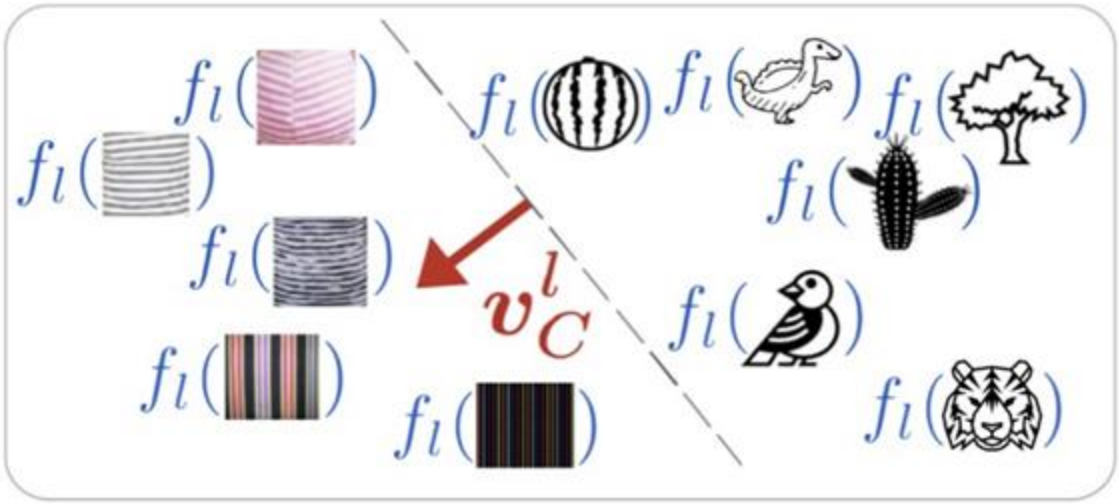
Step 2: Compute Concept Activation Vector (CAV)

Examples of the concept “stripes”

Random examples



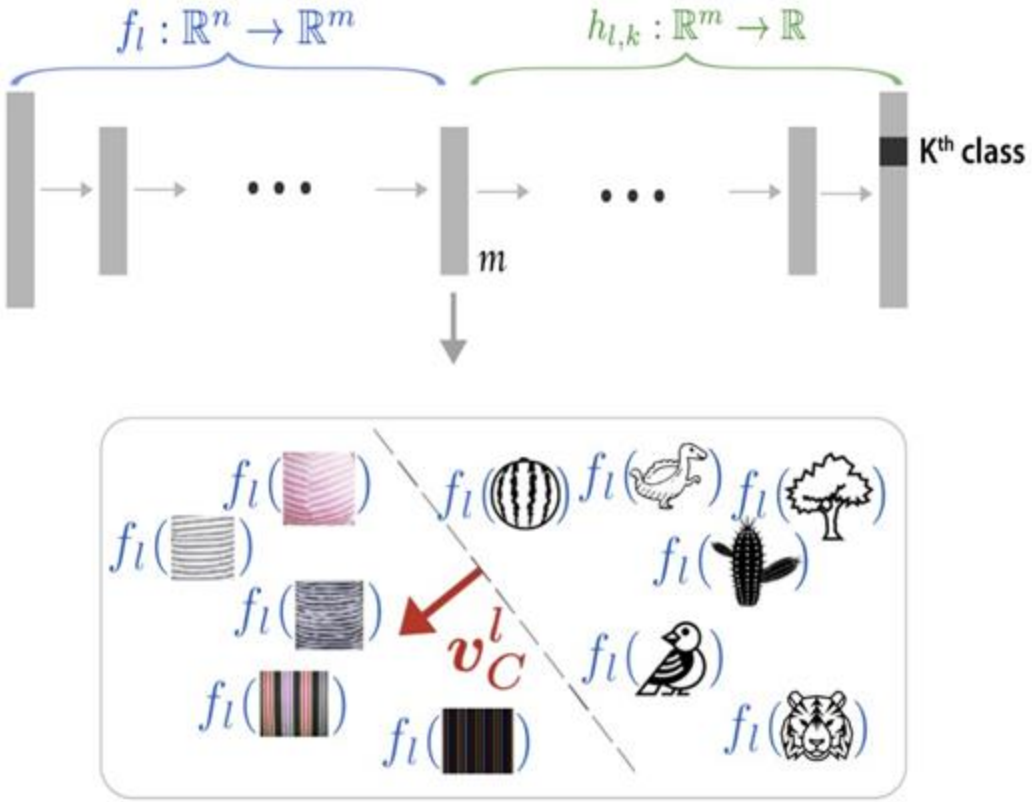
- Consider representations at layer l of all the positive and negative concept examples
- Train a linear classifier to separate positive from negative
- CAV: Vector orthogonal to the decision boundary



Step 3: Compute Conceptual Sensitivity Score

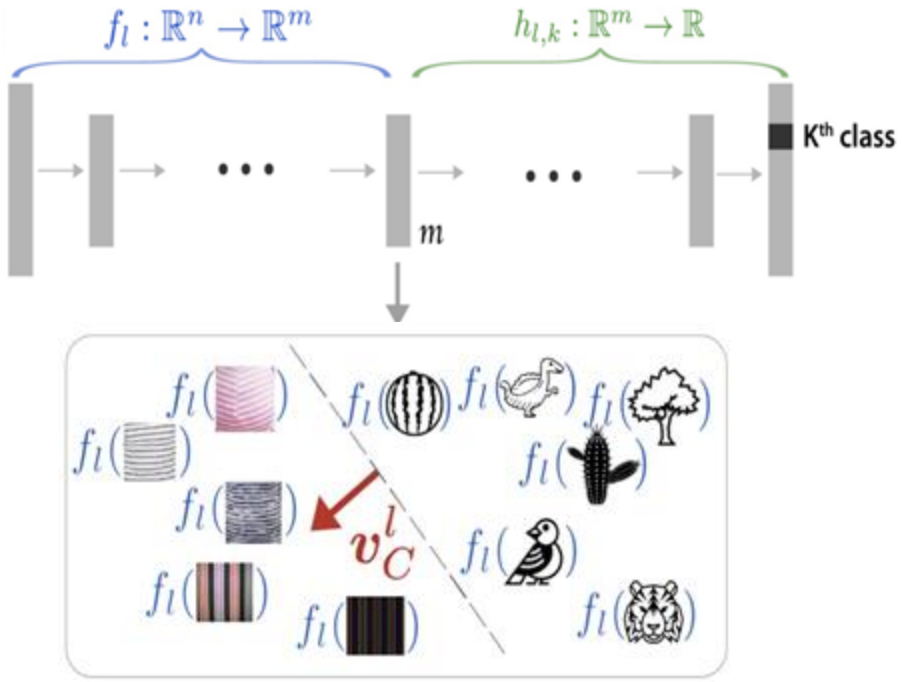
- Recall: in saliency maps, sensitivity of output to an input feature/pixel is determined by the gradient w.r.t. input x
- v_C^l is the unit CAV vector for concept C in layer l
- Conceptual sensitivity of output class k to concept C equals directional derivative:

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



Step 4: Testing with CAV (TCAV)

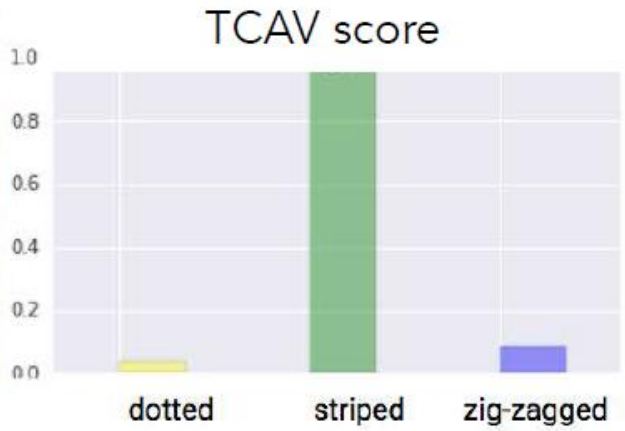
- Let X_k be the set of all training inputs with label k
- Goal: Understand how a model f's prediction for class k is sensitive to a given concept C
- TCAV score: Fraction of k-class training inputs whose l-layer activation vector was positively influenced by concept C



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

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

TCAV Illustration



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

striped CAV →

Directional derivative with CAV

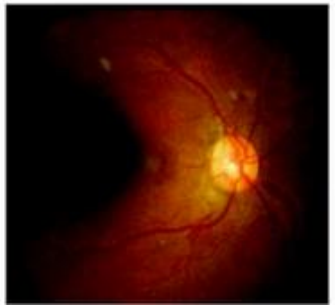
TCAV Evaluation

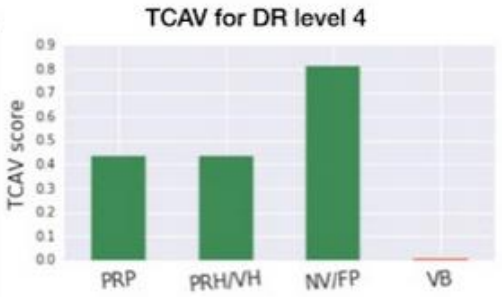
1. Sanity check experiment

cab image cab image with caption

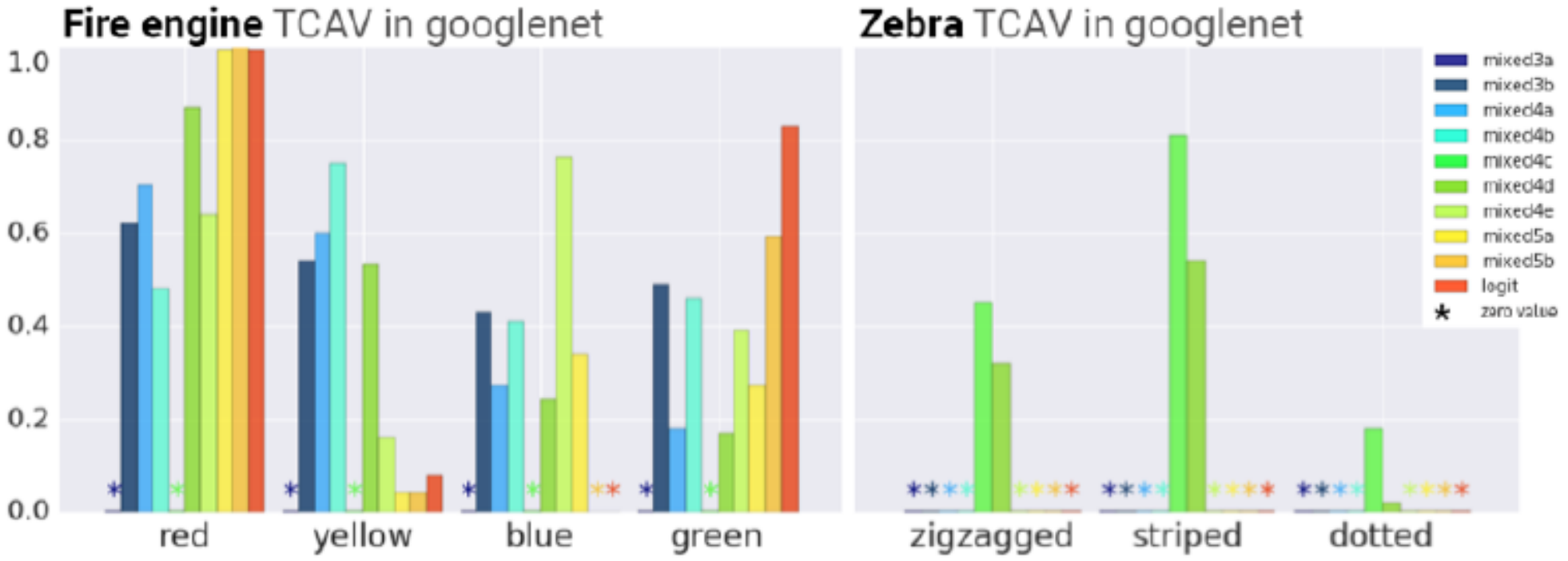
2. Biases in Inception V3 and GoogleNet

3. Domain expert confirmation from Diabetic Retinopathy





TCAV Evaluation



Explainability

- Previously: Feature Attribution Methods
  - LIME (Local Interpretable Model-agnostic Explanations) algorithm
  - SHAP methods based on cooperative game theory
  - Saliency Maps (different versions)
  - Formal guarantees for feature attribution methods

- Today's recap:
  - Counterfactuals
  - Representation-based explanations

- Next lecture: Data Attribution Methods