Decomposing and Editing Predictions by Modeling Model Computation

Harshay Shah, Andrew Ilyas, Aleksander Mądry

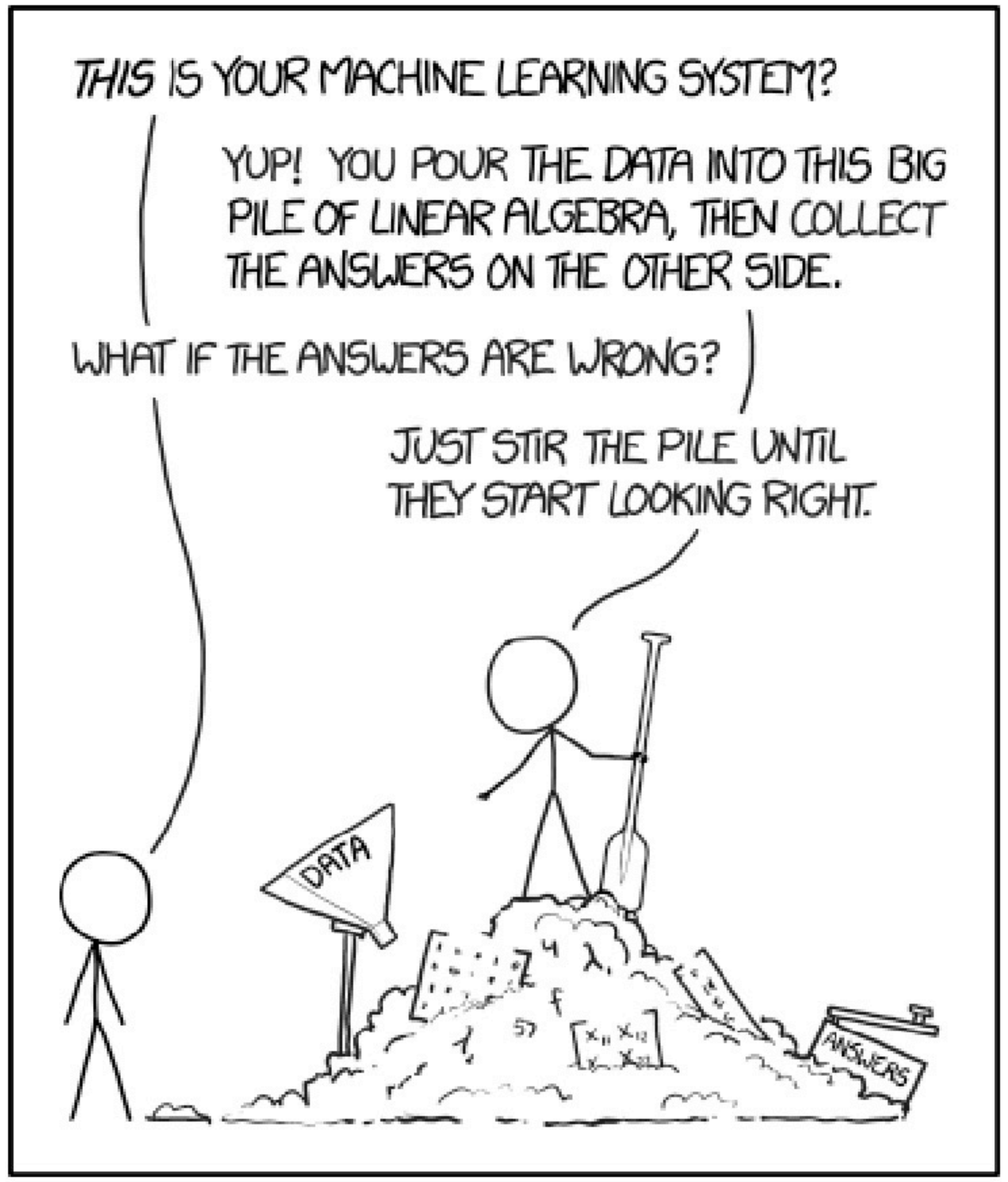
ICML 2024

https://arxiv.org/abs/2404.11534

gradientscience.org/modelcomponents

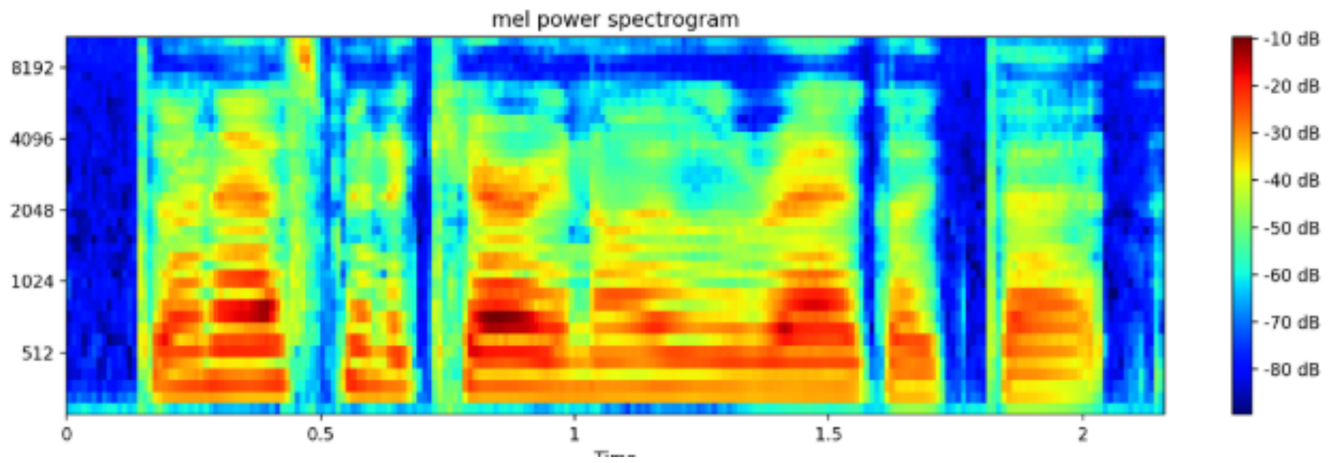
Why study model predictions?

Tinker your ML pipeline



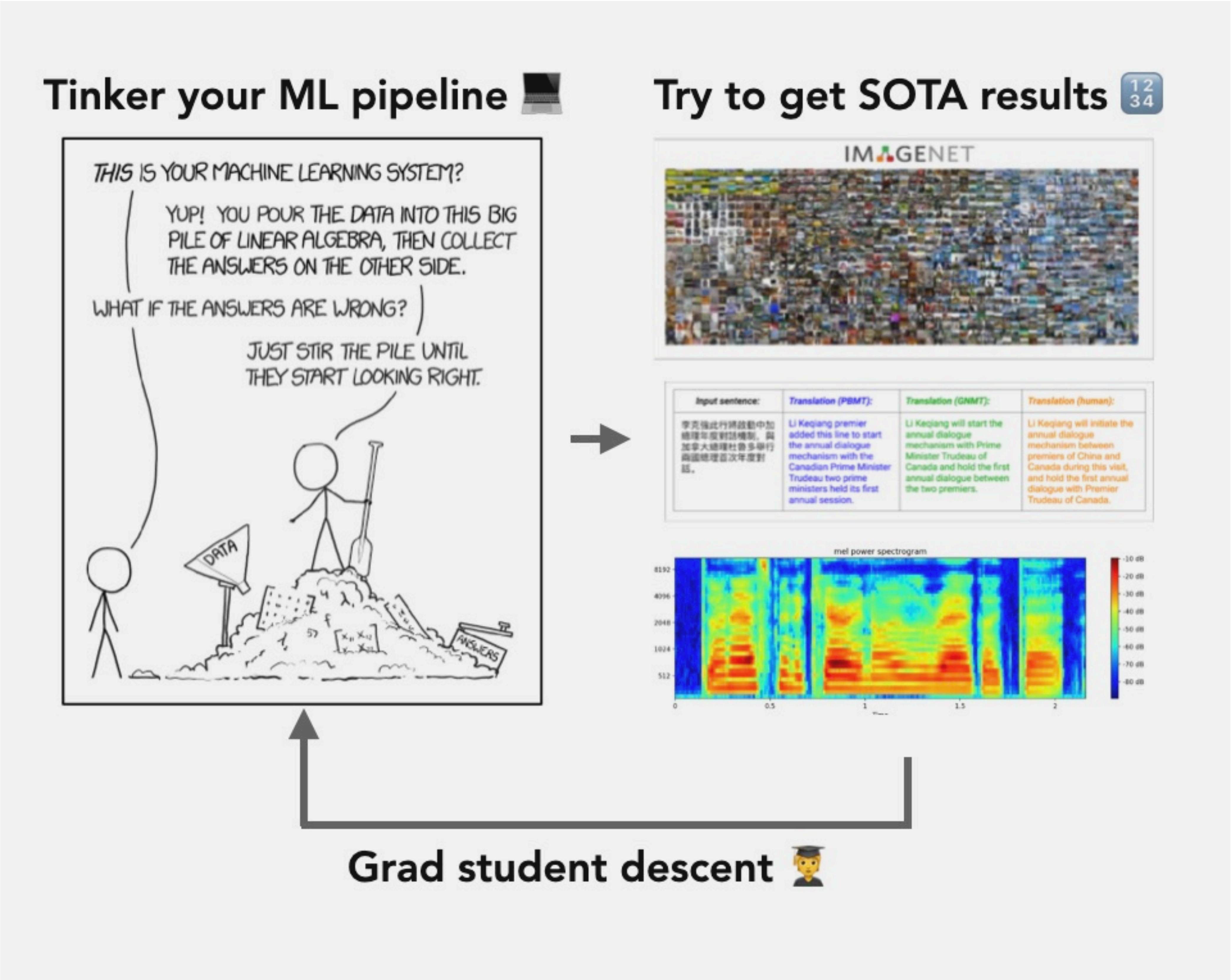
Try to get SOTA results

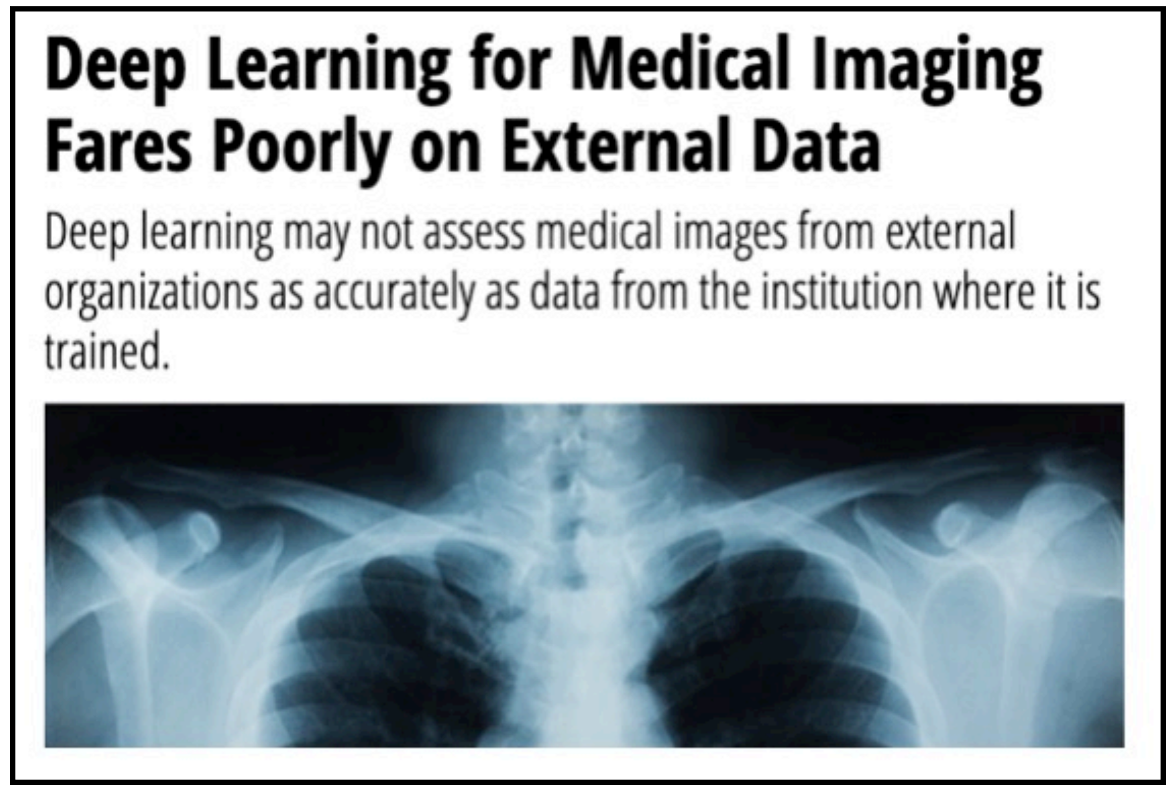
Input sentence:	Translation (PBMT):	Translation (GNMT):	Translation (human):
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.



Repeat

Why study model predictions?




Deep Learning for Medical Imaging Fares Poorly on External Data

Deep learning may not assess medical images from external organizations as accurately as data from the institution where it is trained.

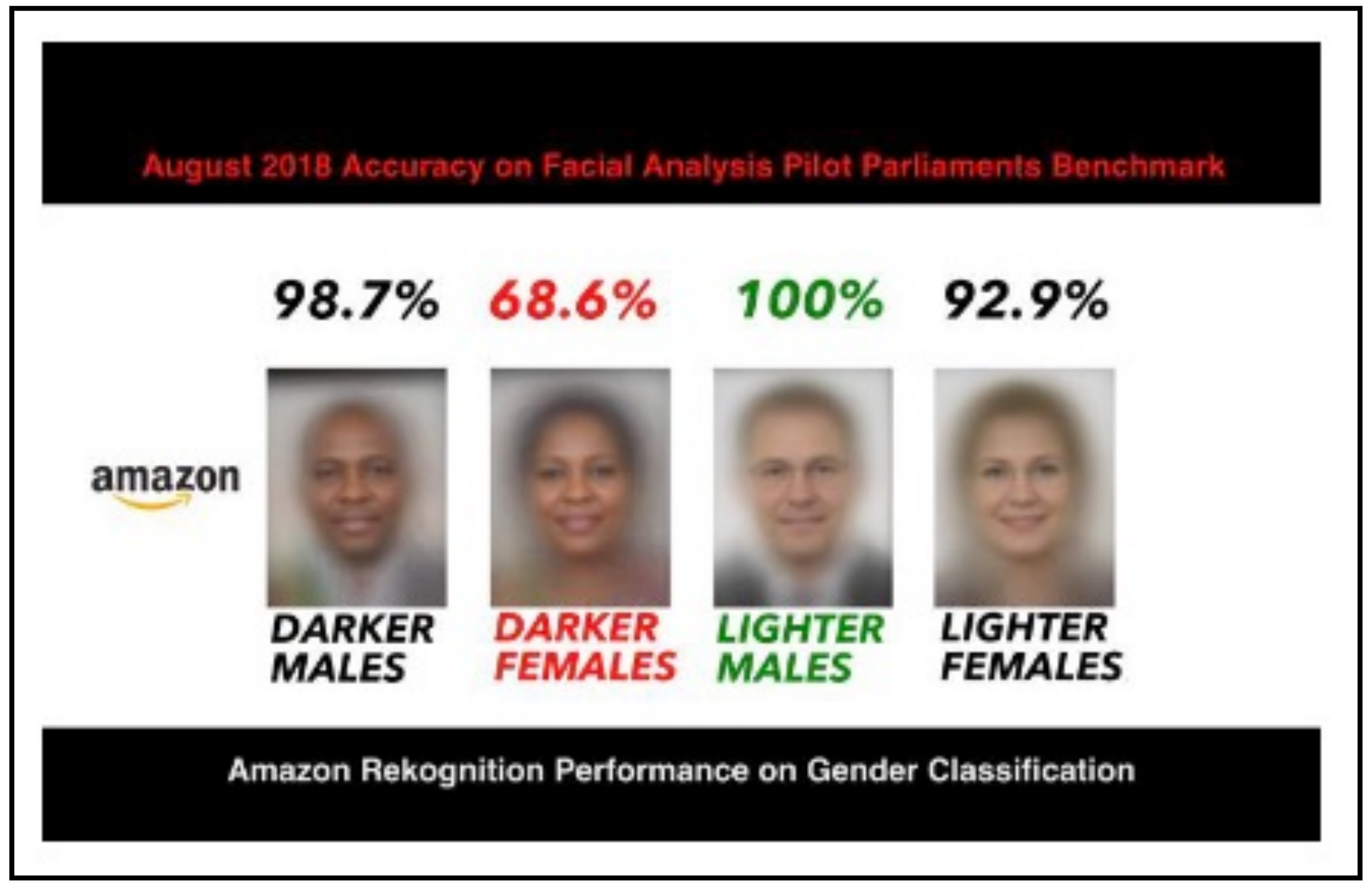

Tesla hit parked police car 'while using Autopilot'

30 May 2018

Share

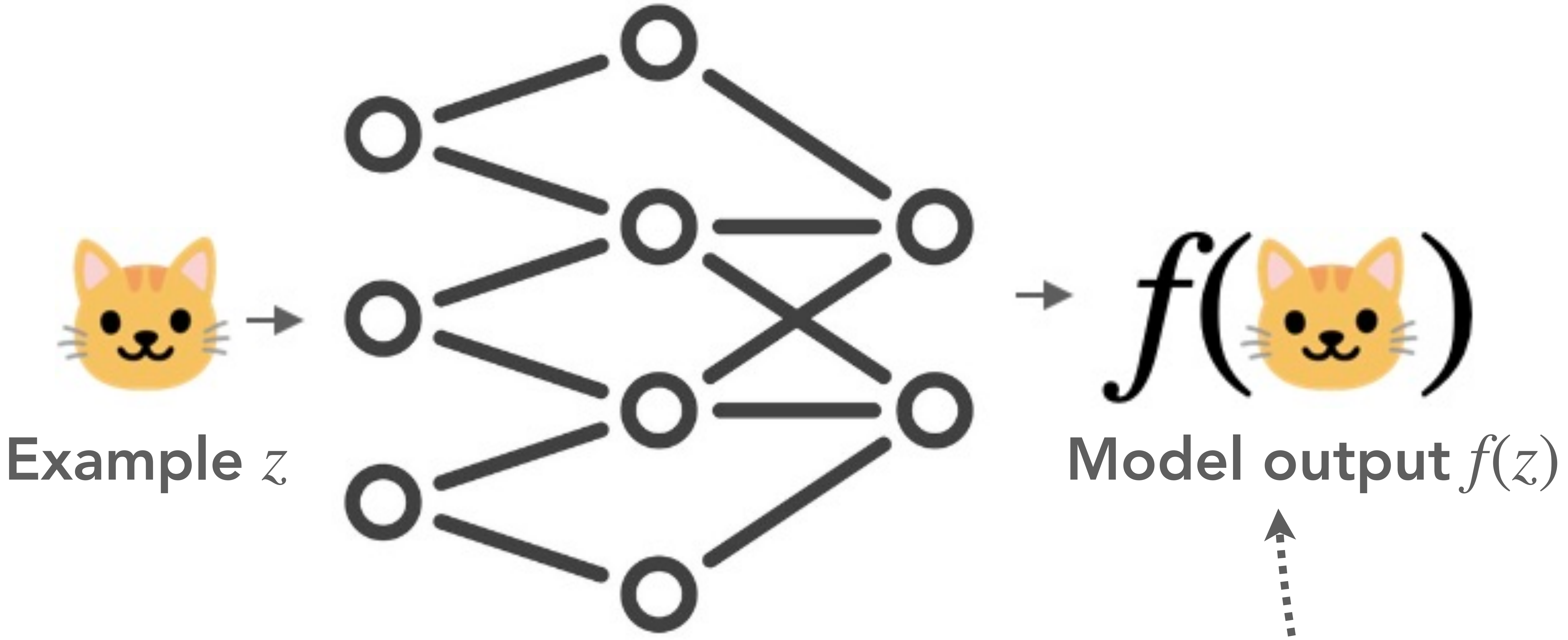
LAGUNA BEACH POLICE DEPARTMENT

A number of Tesla vehicles have been involved in crashes.



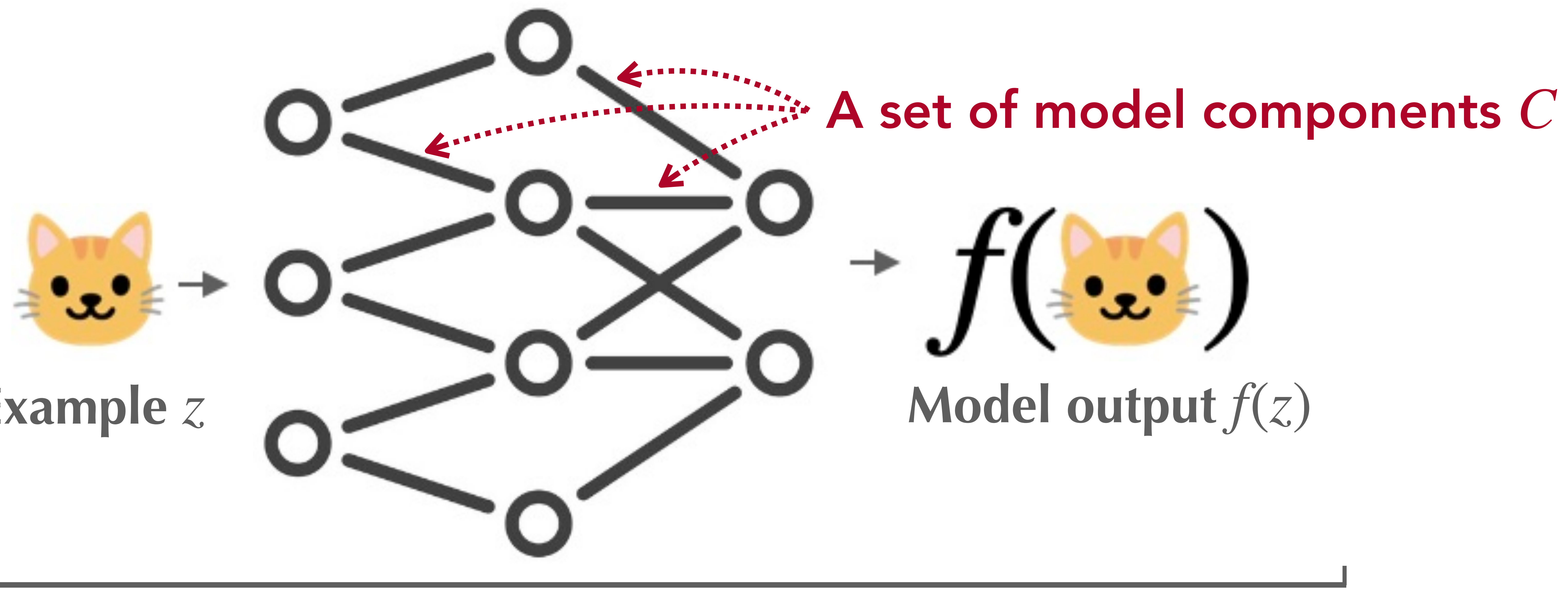
Core issue: We don't understand **how** models internally turn examples into predictions

Models as computation graphs



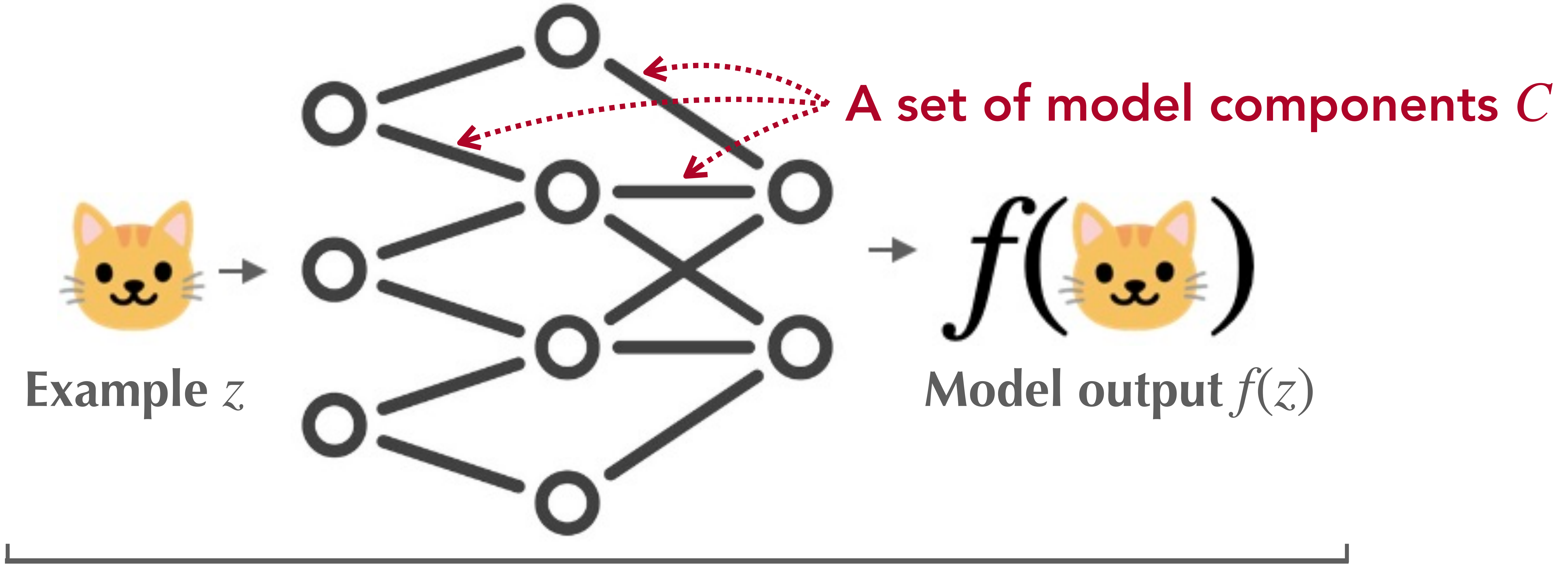
Any metric that quantifies "correctness"
e.g., cross-entropy loss, correct-class confidence., etc

Models as computation graphs



Model f as a computation graph over model components

Models as computation graphs



Model f as a computation graph over model components

Examples of model components in common model architectures

Convolution filters in ResNet models	Weight vectors in MLPs
Attention heads & MLPs in Transformers	Coefficients in linear models

Models as computation graphs

High-level question

Can we somehow understand how model components collectively turn examples into predictions?

Examples

Convolution filters in ResNet models

Weight vectors in MLPs

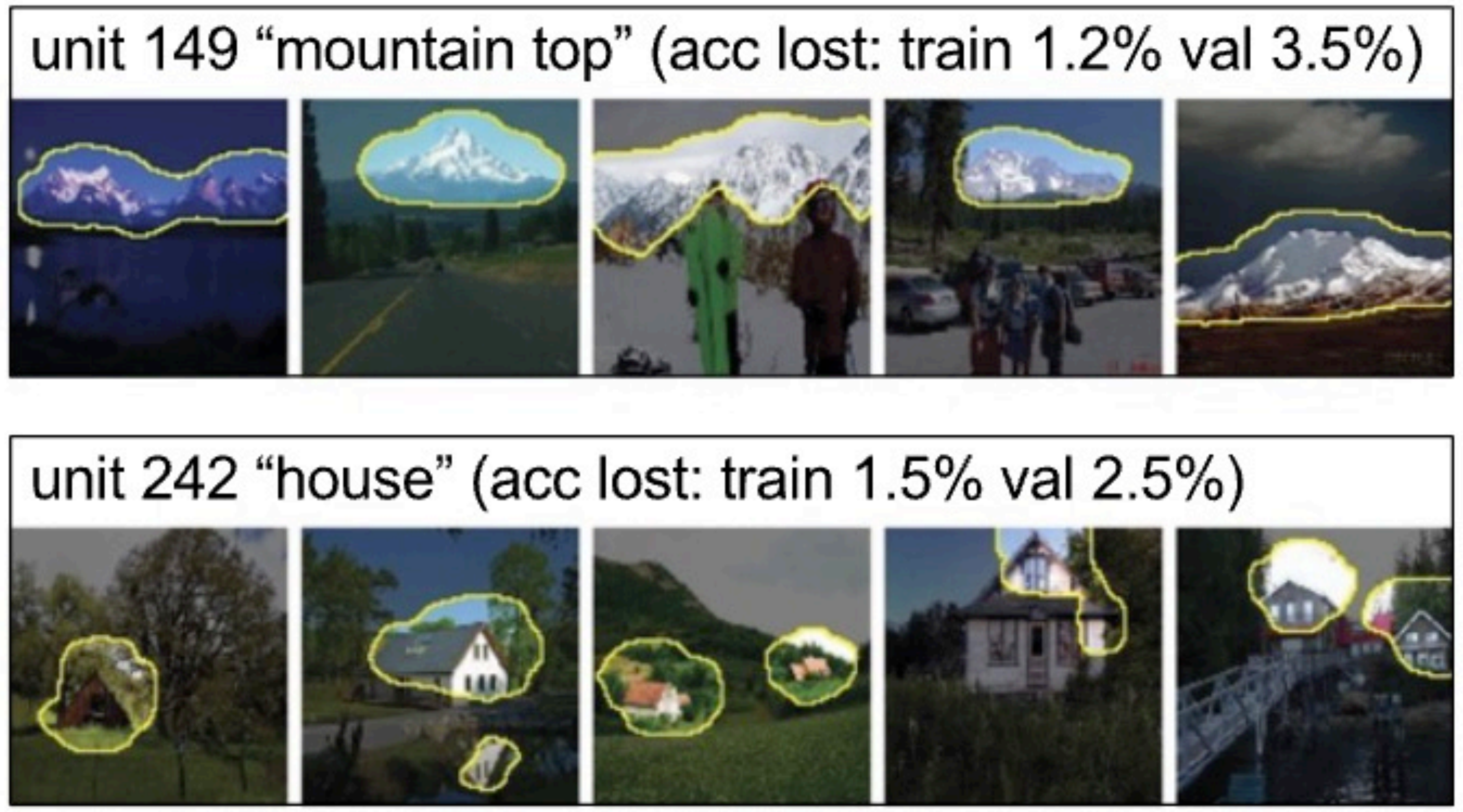
Attention heads & MLPs in Transformers

Parameters in linear models

Background: interpreting model components

Vision models

Convolution filters learn to detect curves and frequency [Cammarata et al. 2020]

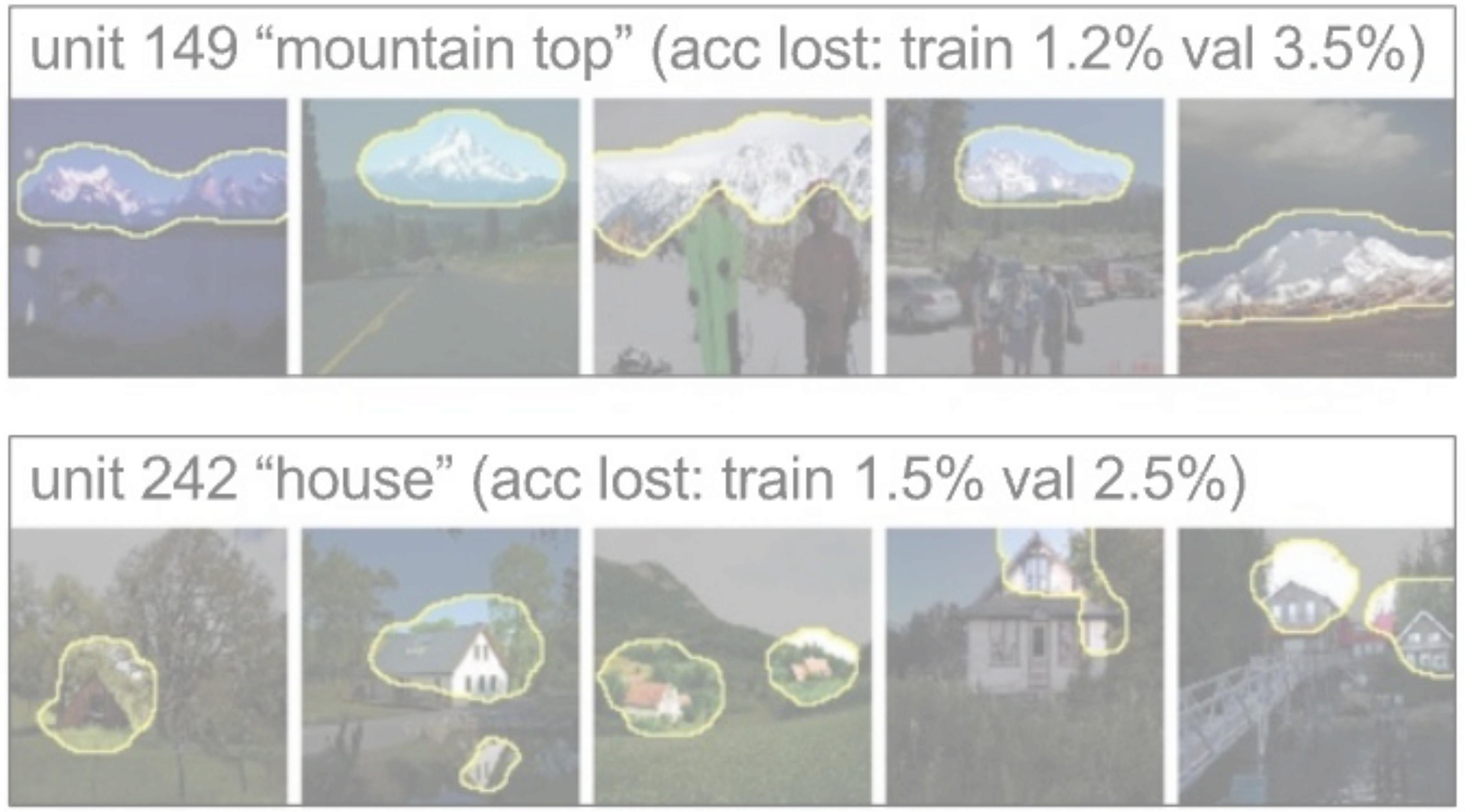


Convolution filters in deeper layers detect high-level concepts [Bau et al. 2020]

Background: Interpreting model components

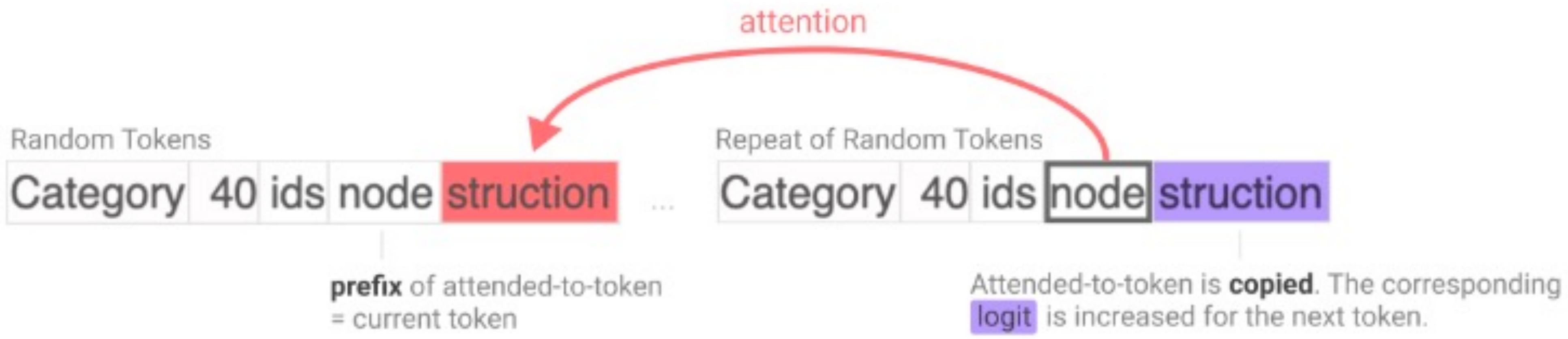
Vision models

Convolution filters learn to detect curves and frequency [Cammarata et al. 2020]

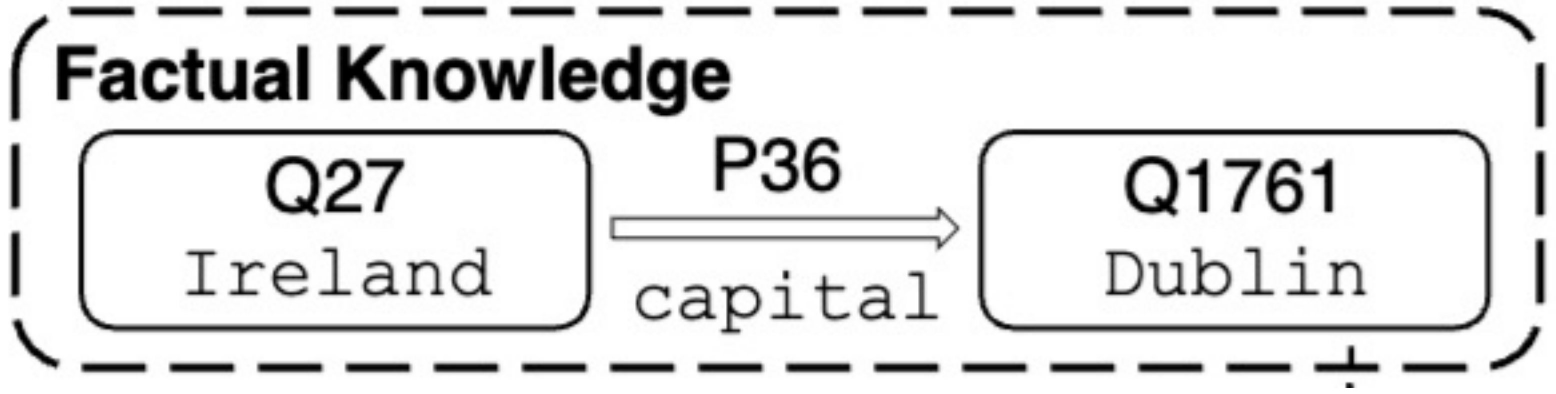


Convolution filters in deeper layers detect high-level visual concepts [Bau et al. 2020]

Language models



Induction heads in transformers [Olsson et al. 2022]



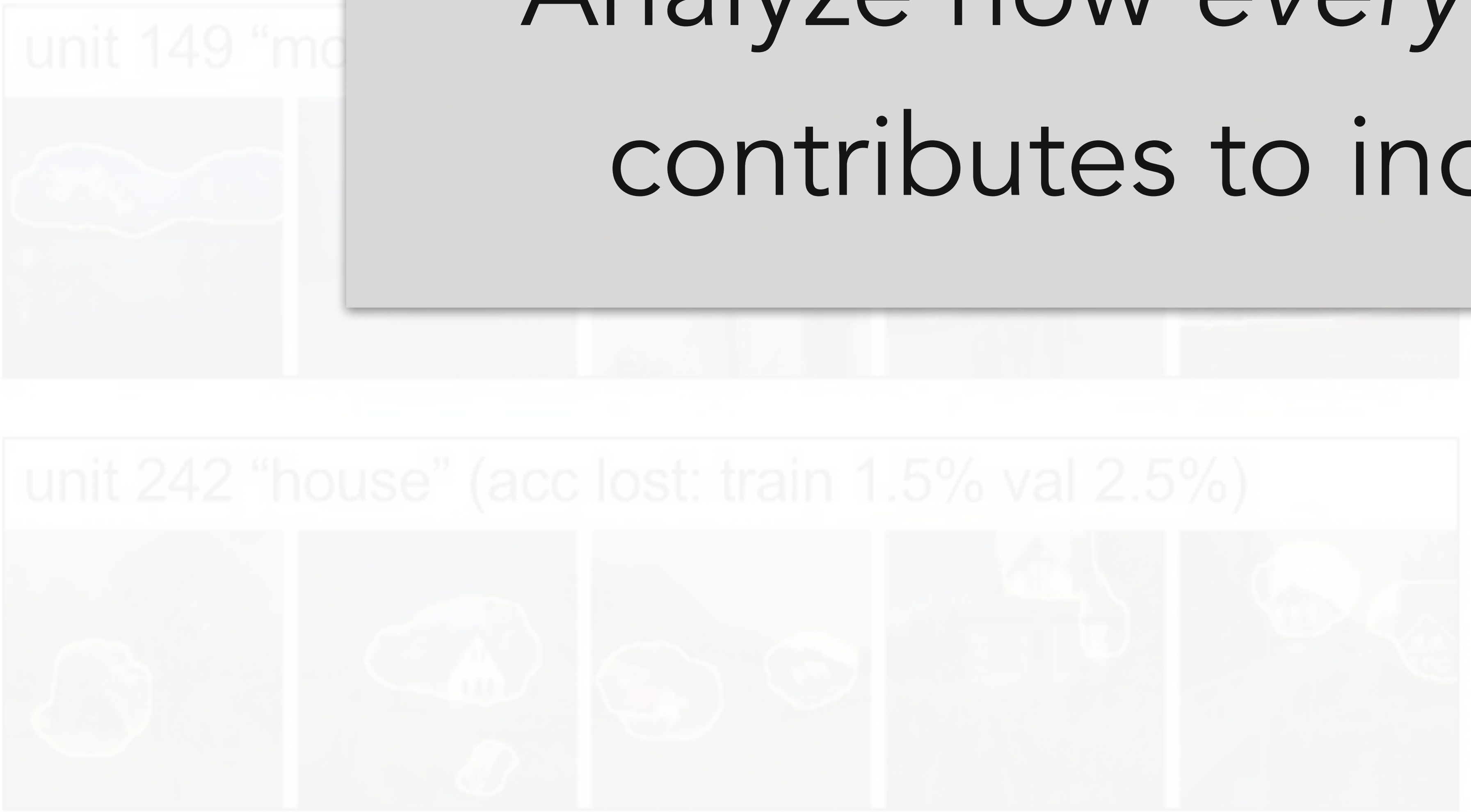
Knowledge neurons encode factual knowledge [Dai et al. 2021]

"Duplicate token head", "Name-mover head", "Backup head", "ML Tea head"... 🤔
[Wang et al. 2022]

Background: interpreting model components

Vision models

Convolutio ...
freque ...



Convolution filters in deeper layers detect high-level visual concepts [Bau et al. 2020]

Language models

Induction heads in transformers [Olsson et al. 2022]

Knowledge neurons encode factual knowledge [Dai et al. 2021]

"Duplicate token head", "Name-mover head", "Backup head", "ML Tea head"... 🤔 [Wang et al. 2022]

Our goal

Analyze how *every* model component $c \in C$ contributes to individual predictions $f(\cdot)$

Our work

Component attribution framework

Decompose any prediction into "contributions" from every model component

COAR: Component Attribution via Regression

A general method for efficient and accurate component attribution

COAR-Edit: Model editing using component attributions

Edit model behavior by ablating a targeted subset of components

The component attribution framework

Main idea

If we can "understand" how all model components shape a prediction

↓

we should be able to estimate how predictions <u>change</u> in response to interventions to one or more model components

The component attribution framework

Main idea

If we can "understand" how all model components shape a prediction,

↓

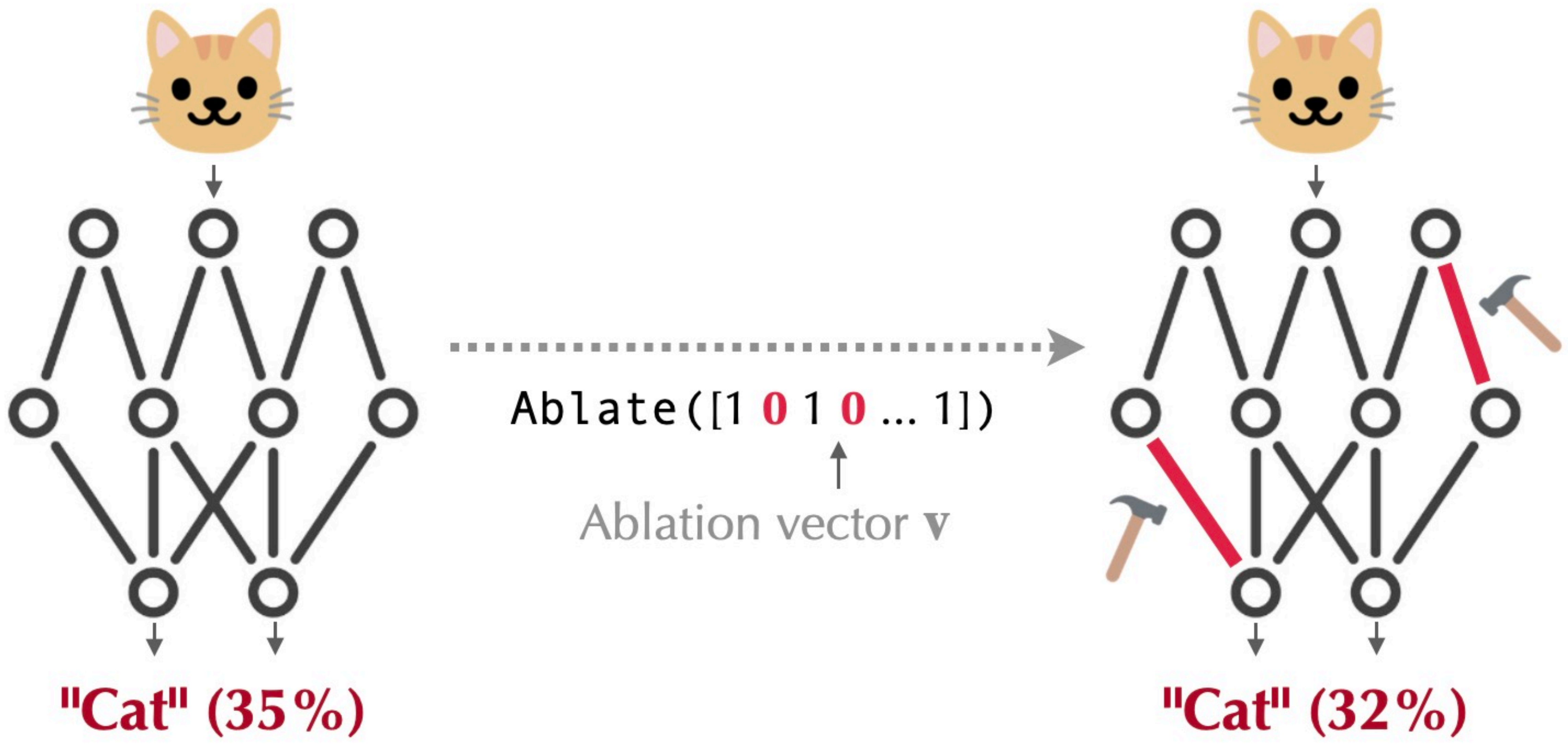
we should be able to estimate how interventions to model components

<u>change</u> model predictions

Component ablations as interventions

A **component ablation** intervenes on the *parameters* corresponding to one or more model components.

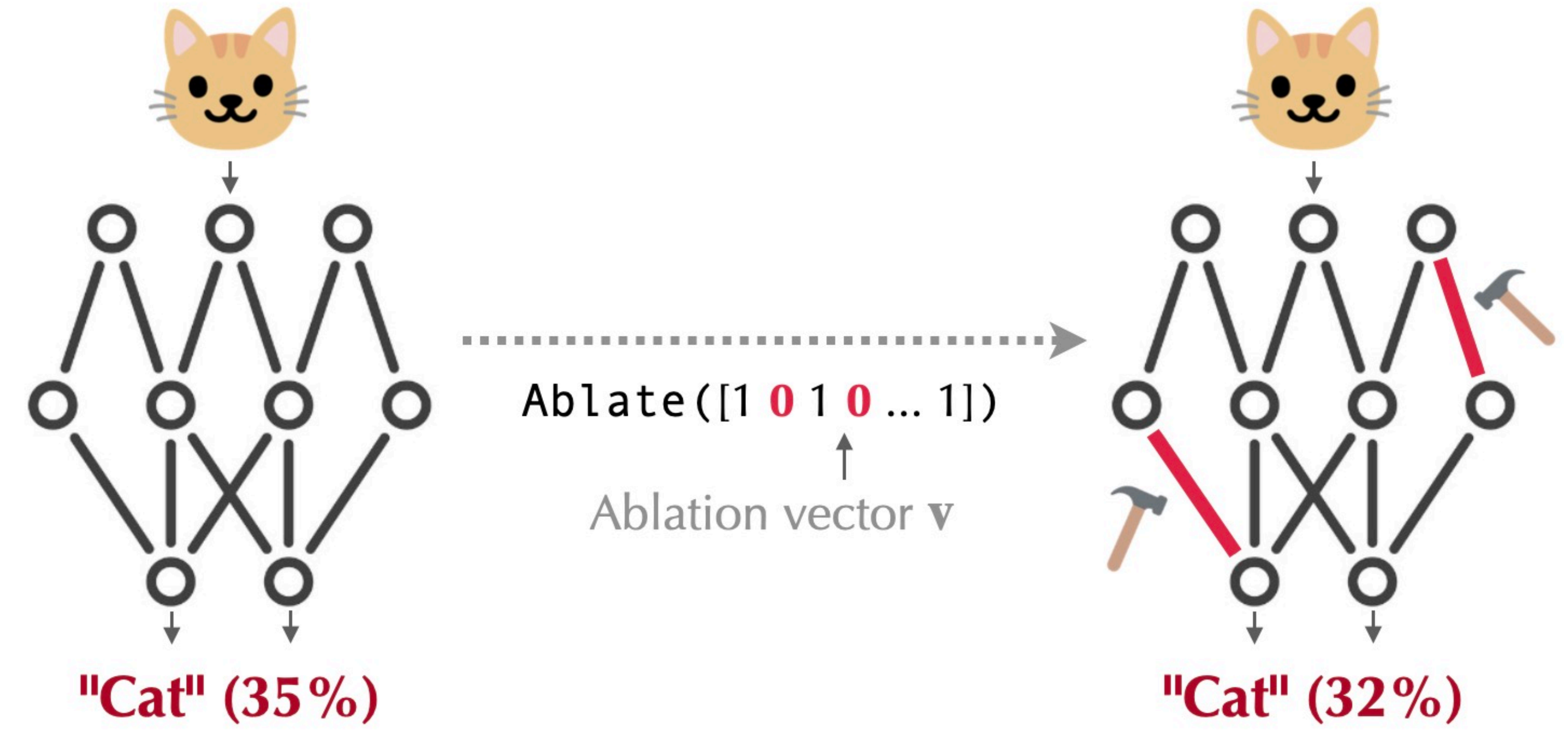
For instance, zeroing out or adding noise



The component attribution framework

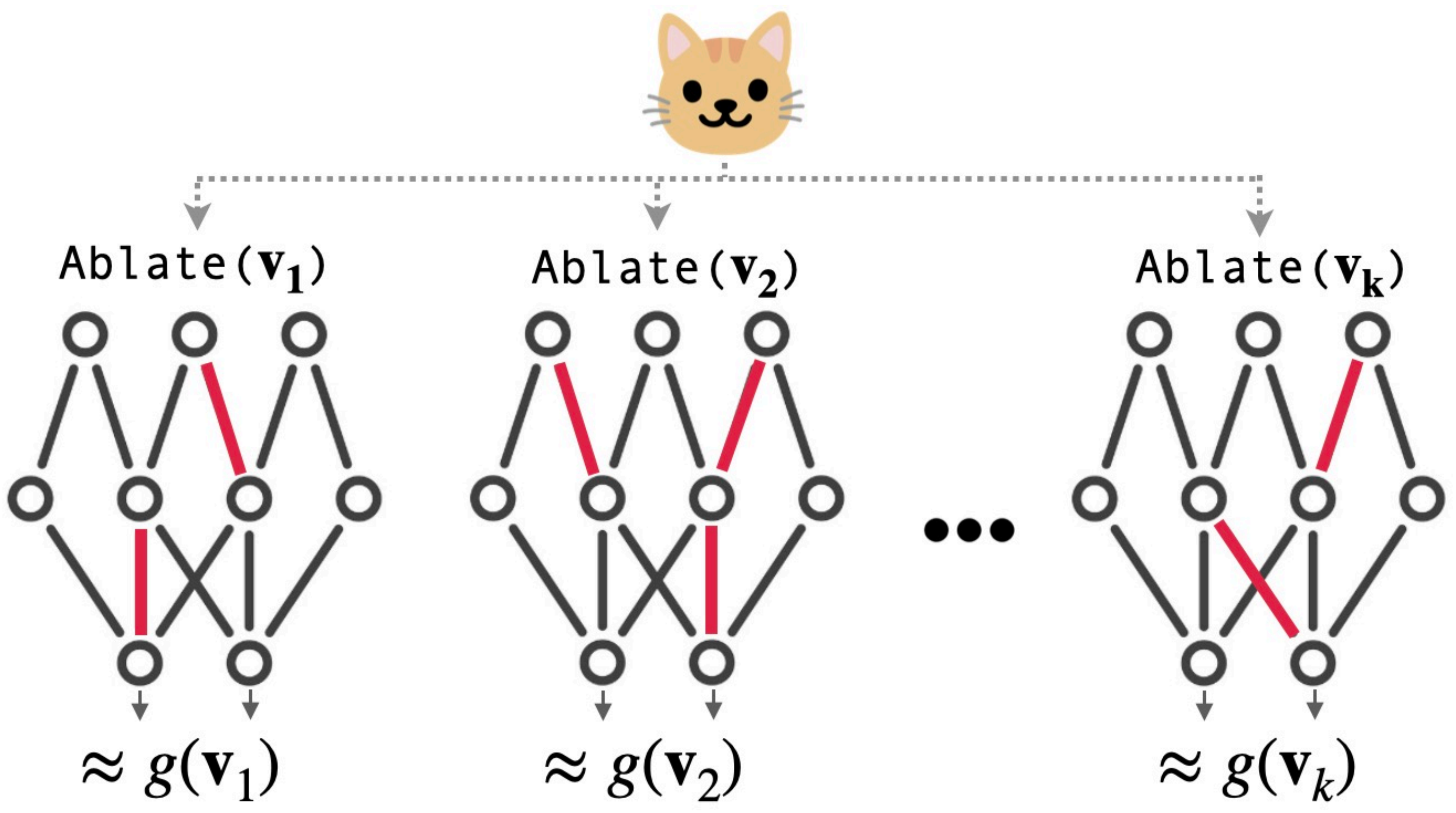
Component ablations as interventions

A **component ablation** intervenes on the *parameters* corresponding to one or more model components.



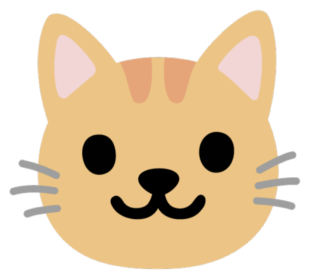
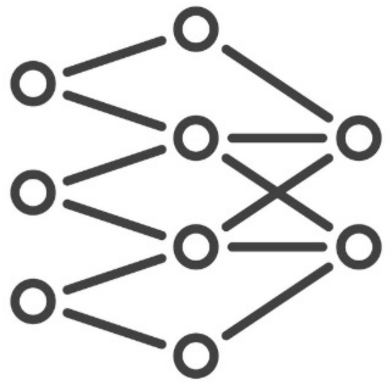
Component attribution

A **component attribution** g takes as input an ablation vector v and estimates the effect of the component ablation on a given model prediction.



Formalizing component attribution

Fix:

Example z	Trained model f	Set of components C
e.g., from ImageNet	e.g., a ResNet50	e.g., conv filters in all layers
		`layer7.block3.conv[42]`

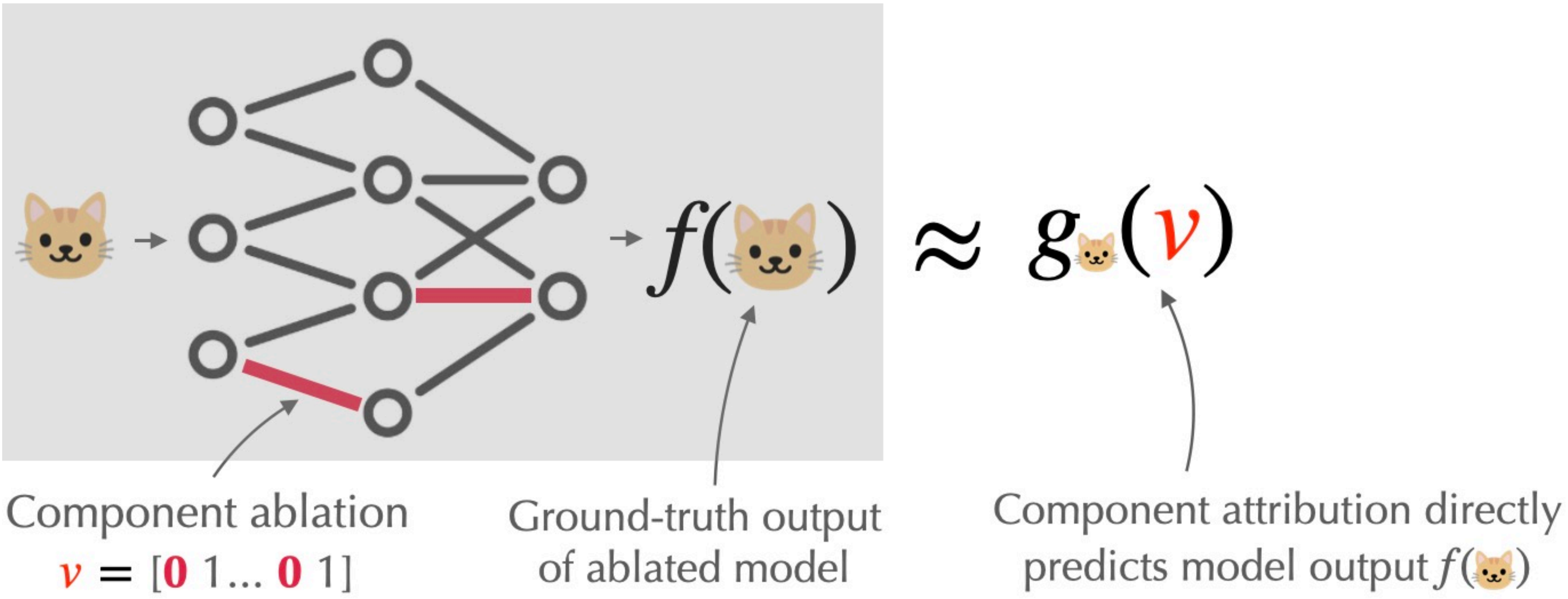
For **any** component ablation $v \in \{0,1\}^{|C|}$

1. Using v, apply component ablation to the model f

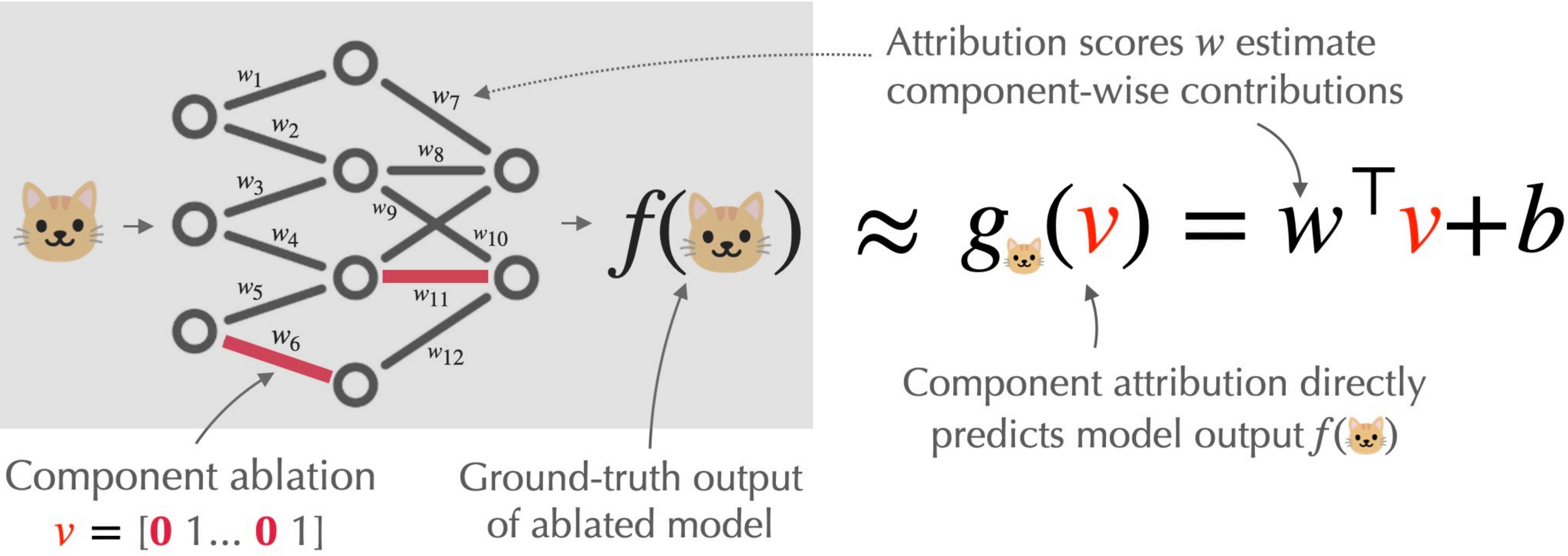
2. Evaluate output of ablated model on example z to get $f(z, v)$

Goal: Given (any) component ablation v, estimate $f(z, v)$ (i.e., without intervening)

Formalizing component attribution

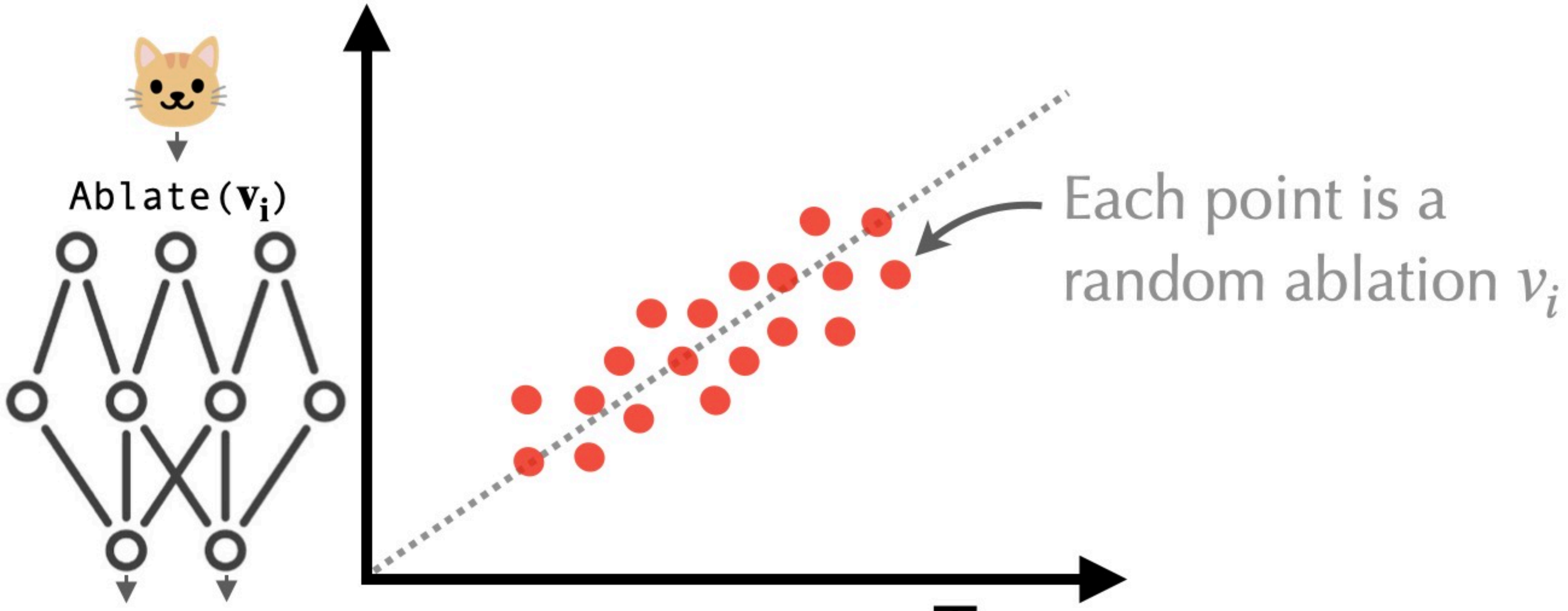


Formalizing component attribution



Formalizing component attribution

Next: We want to estimate component attributions that accurately predict how component ablations change model predictions



$$g(\mathbf{v_i}) = b + \mathbf{w}^\top \mathbf{v_i}$$

Component attribution

Attribution vector w quantifies component-wise contributions

Our work

Component attribution framework

Decompose any prediction into "contributions" from every model component

COAR: <u>Co</u>mponent <u>A</u>ttribution via <u>R</u>egression

A general method for efficient and accurate component attribution

COAR-Edit: Model editing using component attributions

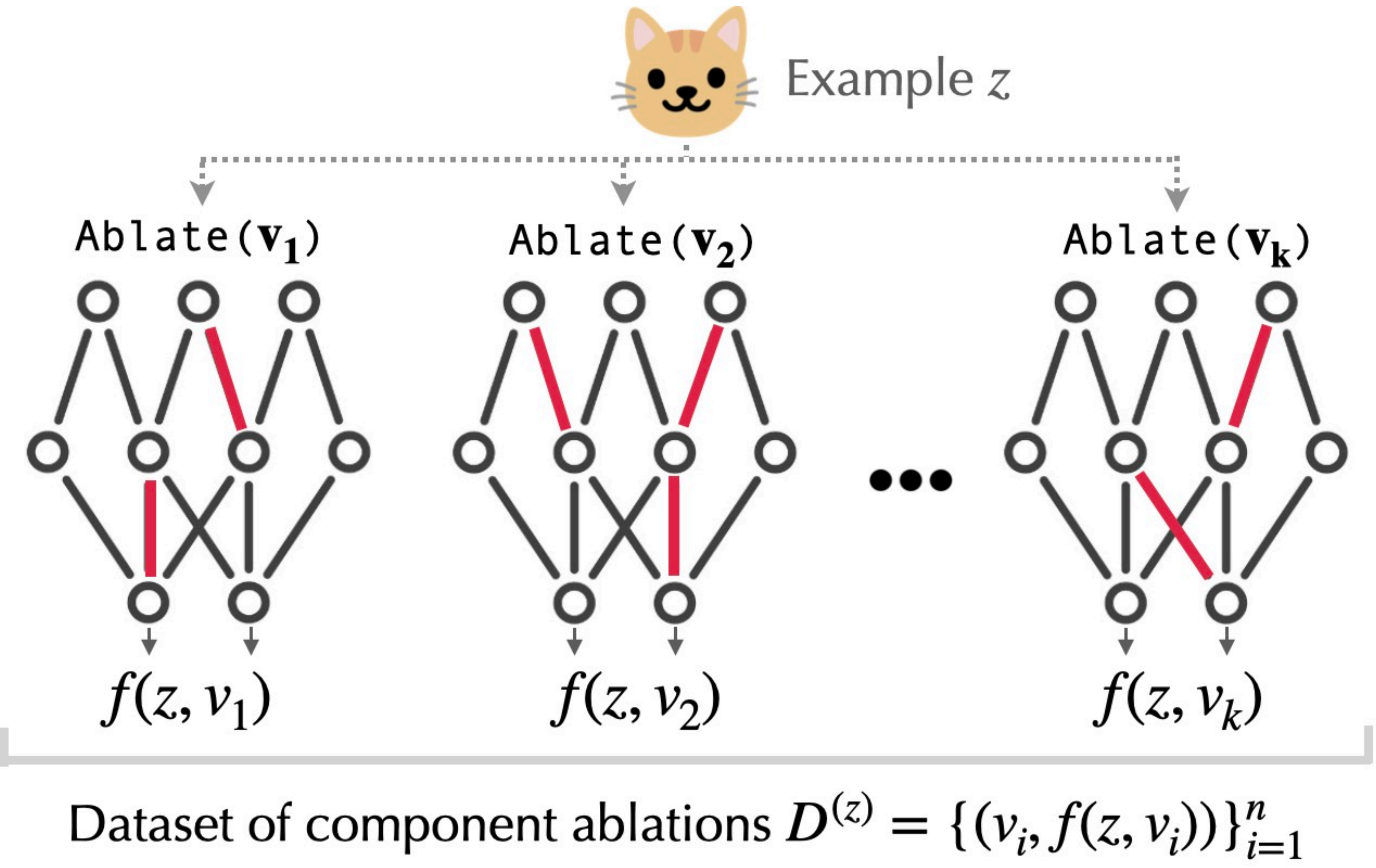
Edit model behavior by ablating a targeted subset of components

COAR: Component Attribution via Regression

Cast component attribution into a **supervised learning** problem in two steps

Step 1/2

Construct a dataset of component ablations by ablating random subsets of components and recording both the ablations and the ablated model's outputs for each example of interest.



COAR: Component Attribution via Regression

Cast component attribution into a **supervised learning** problem in two steps

Step 2/2

Fit a linear regression model that maps an ablation vector v_i to the ablated models' output $f(z, v_i)$. The weights (w, b) of this linear model serve as our component attribution $g^{(z)}(v) = w^\top v + b$

Ground-truth output of ablated model

$$(w^{(z)}, b^{(z)}) = \arg\min_{w,b} \sum_{D^{(z)}} \left(f(z, v_i) - v_i^\top w - b\right)^2$$

Dataset of component ablations

Attribution-based estimate

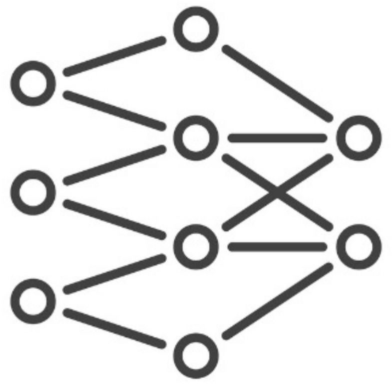
COAR: Component Attribution via Regression

Does COAR learn **accurate** component attributions?

Setup

Example z
from ImageNet

Model f
ImageNet-trained ResNet50



Components C
22,720 conv filters

`layer*.block*.conv*`

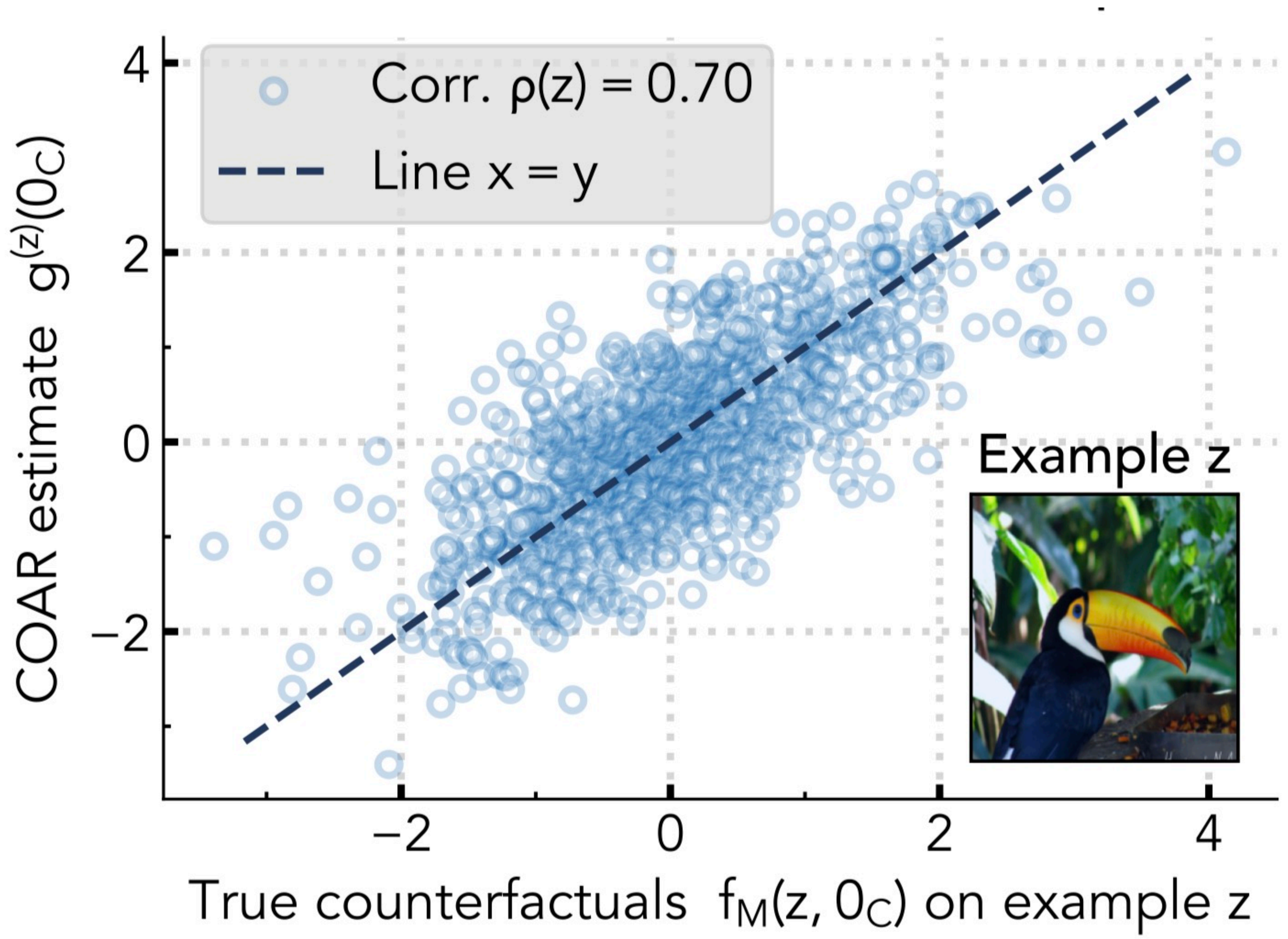
Evaluating component attributions $g^{(z)}$

1. Sample an (unseen) <u>random</u> ablation vector v
2. Check if the attribution-based estimate $g^{(z)}(z)$ predicts ground-truth output $f(z, v)$

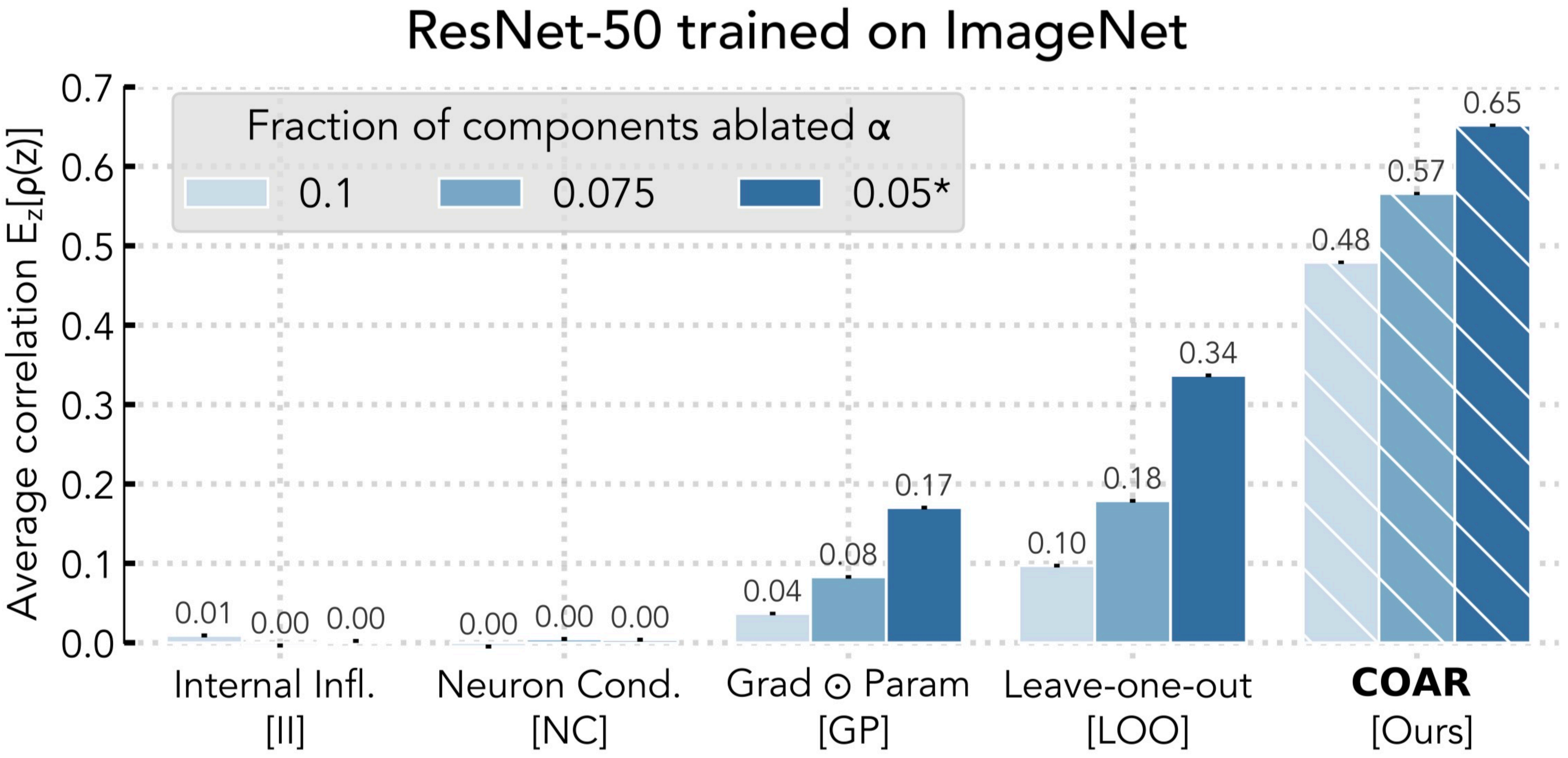
COAR: Component Attribution via Regression

Does COAR learn **accurate** component attributions?

ResNet-50 trained on ImageNet



COAR: Component Attribution via Regression



COAR: Component Attribution via Regression

ResNet-50 trained on ImageNet

Results consistent across

Architectures: MLPs, CNNs, Transformers

Language models: GPT-2, Phi-2, Llama-7b

Datasets: ImageNet, CIFAR-10, TinyStories, BoolQ

Our work

Component attribution framework

Decompose any prediction into "contributions" from every model component

COAR: <u>Co</u>mponent <u>A</u>ttribution via <u>R</u>egression

A general method for efficient and accurate component attribution

COAR-Edit: Model editing using component attributions

Edit model behavior by ablating a targeted subset of components

COAR-Edit: Model editing using COAR attributions

Component attribution asks

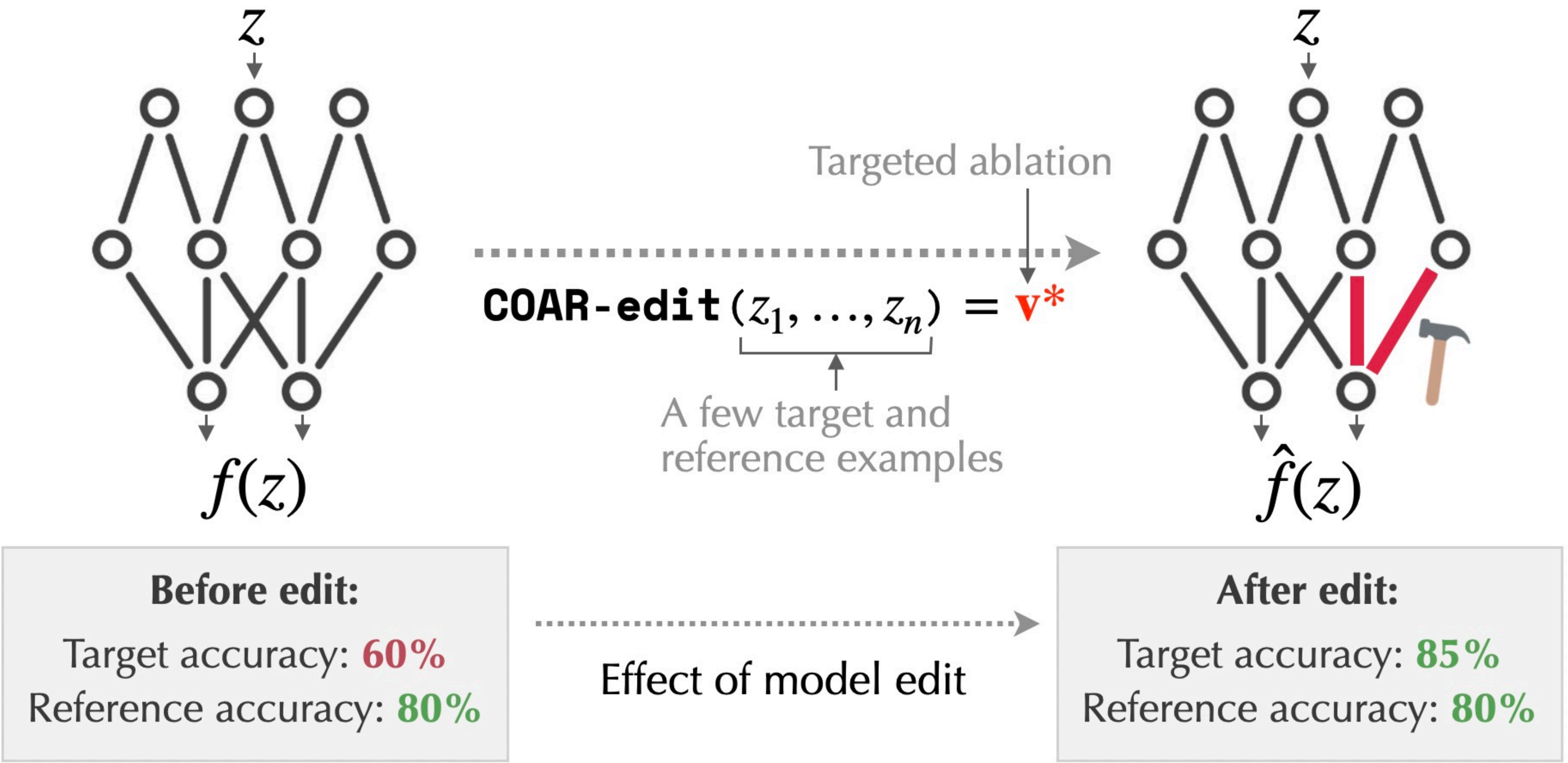
How would model outputs change if we were to ablate a subset of components?

Model editing inverts this to

Which components, when ablated, would change model outputs in a specific way?

COAR-Edit: Model editing using COAR attributions

Goal: perform a model edit that improves performance on **target examples** without degrading performance on **reference examples**



COAR-Edit: Model editing using COAR attributions

Main idea

Use COAR attributions to identify model components that,
when ablated, change model behavior in a targeted manner

No additional training needed ✅ **Sample-efficient** ✅

COAR-Edit: Model editing using COAR attributions

Step 1/3

Compute COAR attributions for target and reference examples

Step 2/3

For every component, quantify its "importance" to target examples *relative* to reference examples with a simple t-test (null: target ~ reference)

Step 3/3

Ablate the bottom-k components with the lowest test statistics to improve model performance on the target examples.

Case study #1: Improving group robustness

Problem

1. Models latch on to spurious correlations in the training dataset

2. At test time, models performance sucks when spurious correlation is absent

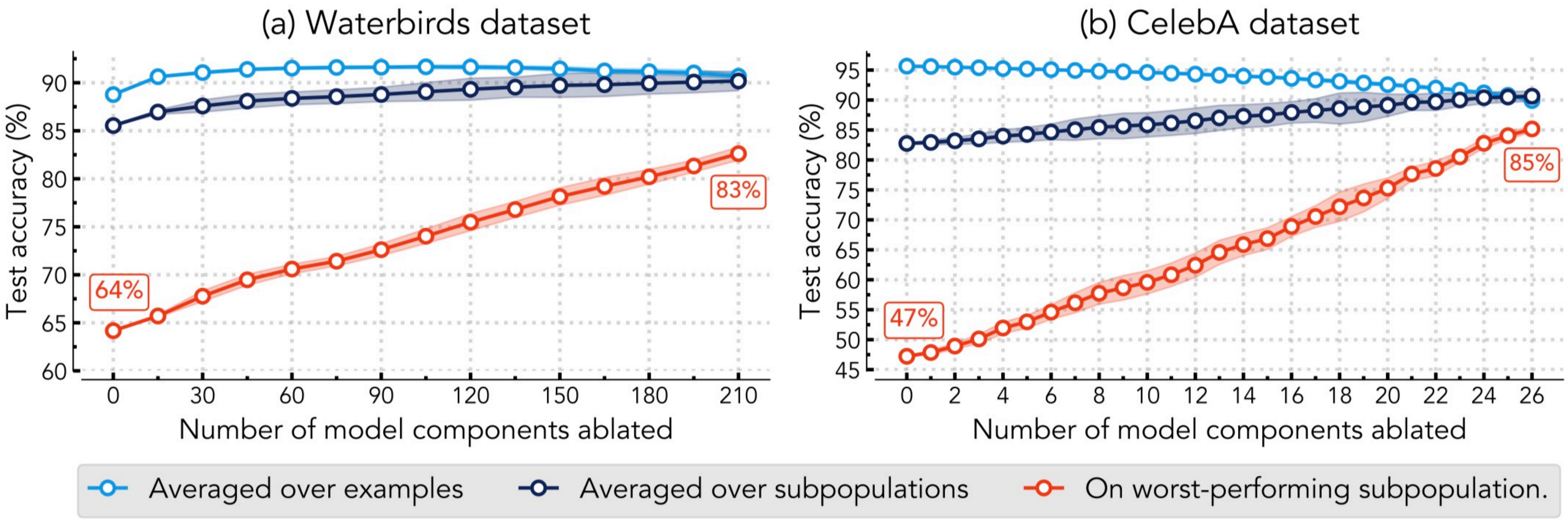
Under-performing minority groups

[Group robustness benchmarks from Sagawa et al. 2020]

Case study #1: Improving group robustness

Applying COAR-Edit

1. **Target examples**: a few examples (~10) from the majority group(s)
2. **Reference examples**: a few examples (~10) from the minority group(s)



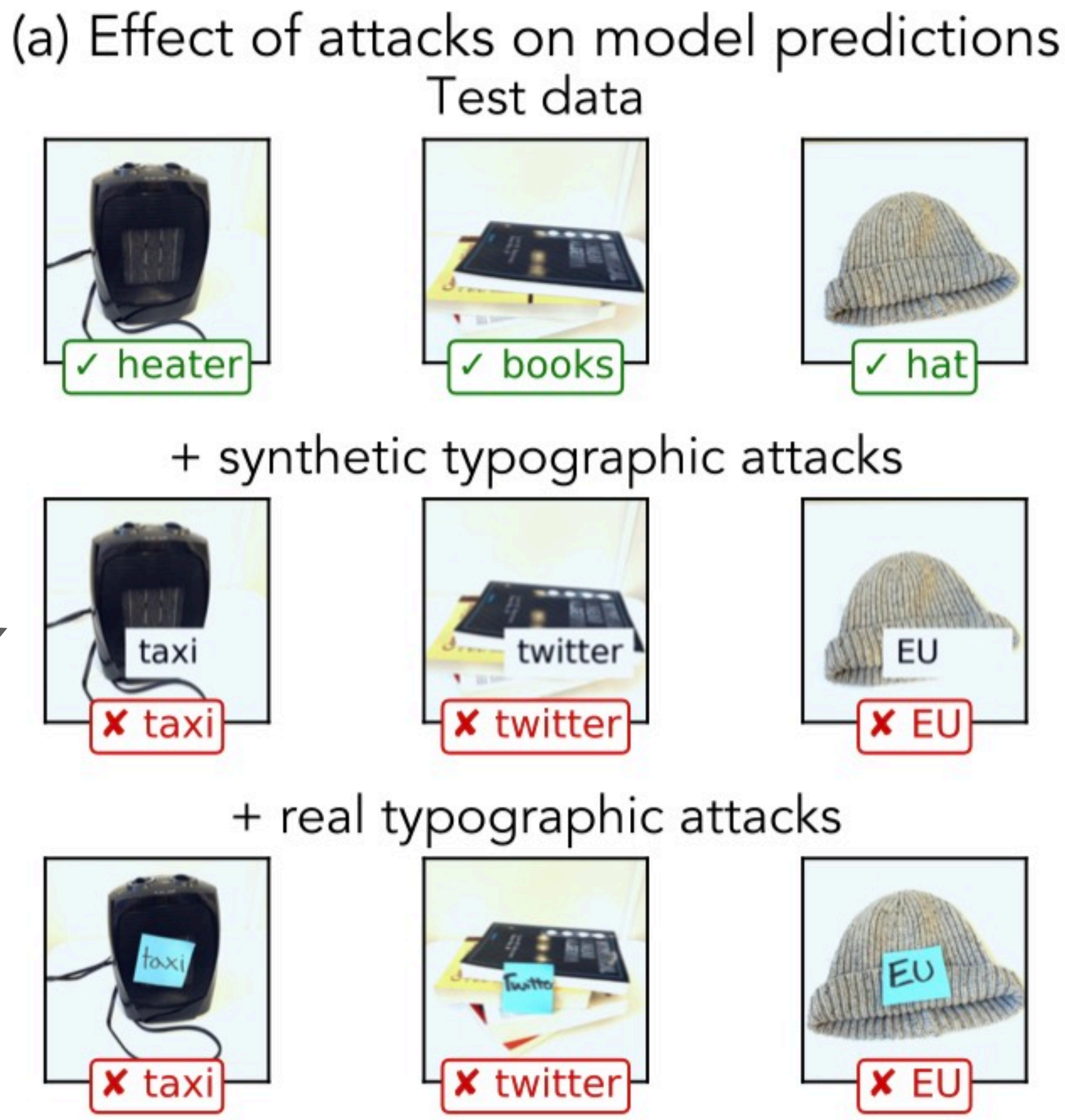
Case study #2: Robustness to typographic attacks

Problem

Zero-shot CLIP classifiers are sensitive to typographic attacks

[Goh et al. 2021]

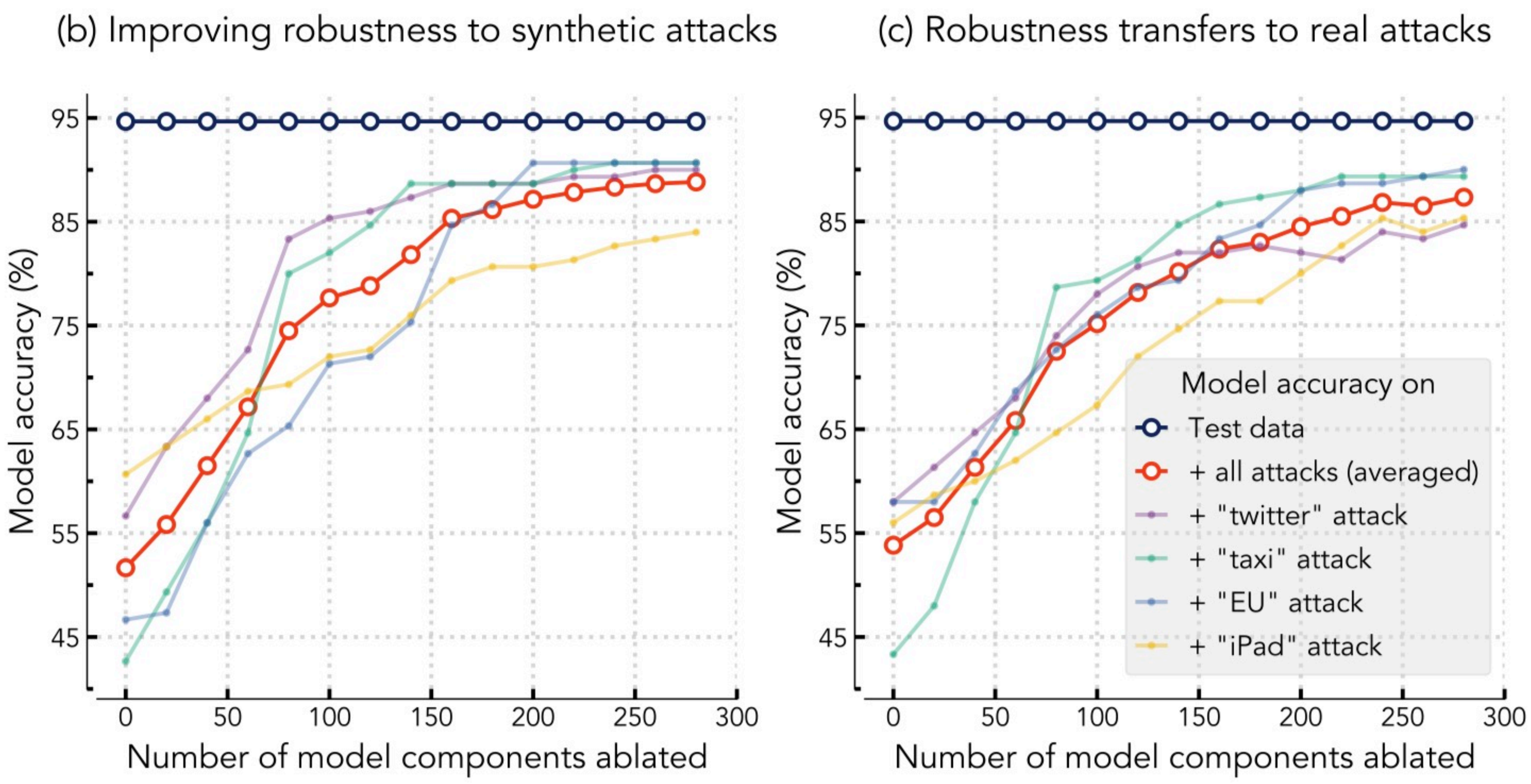
Evaluating a CLIP ViT-B/16 model on images w/ and w/o attacks



Case study #2: Robustness to typographic attacks

Applying COAR-Edit

1. **Target examples**: a few examples (~10) with synthetic typographic attacks
2. **Reference examples**: a few examples (~10) without typographic attacks



Summary

Decomposing and Editing Predictions by Modeling Model Computation

→ Decompose predictions into contributions from every model component

→ How? Use **COAR** to learn component attributions

→ Edit model behavior at the level of examples, subpopulations, and concepts

→ How? Use **COAR-Edit** to identify and ablate a targeted set of model components

Check out our paper for more findings!

`https://arxiv.org/abs/2404.11534`

@harshays_