Decomposing and Editing Predictions by Modeling Model Computation

Harshay Shah [1] **Andrew Ilyas** [1] **Aleksander Mądry** [1]

Abstract

How does the internal computation of a machine learning model transform inputs into predictions? To tackle this question, we introduce a framework called *component modeling* for decomposing a model prediction in terms of its *components*—architectural "building blocks" such as convolution filters or attention heads. We focus on a special case of this framework, *component attribution*, where the goal is to estimate the counterfactual impact of individual components on a given prediction. We then present COAR, a scalable algorithm for estimating component attributions, and demonstrate its effectiveness across models, datasets and modalities. Finally, we show that COAR directly enables effective model editing. Our code is available at github.com/MadryLab/modelcomponents.

1. Introduction

Despite their predictive power, machine learning models remain black boxes. In particular, the internal computation that these models perform to transform inputs into predictions makes it difficult to understand model behavior and, as a result, detect failure modes prior to deployment (Beery et al., 2018; Sheng et al., 2019; Geirhos et al., 2020).

In response to this difficulty, a line of work in ML interpretability aims to shed light on model computation by analyzing *model components*—intuitively "grouped" model parameters such as convolutional filters or attention heads. For example, feature visualization methods (Simonyan et al., 2013; Zeiler & Fergus, 2014; Ghiasi et al., 2022) identify components in vision models that detect visual concepts such as curves (Olah et al., 2020a) and objects (Bau et al., 2020b). Representation-based probes (Alain & Bengio, 2016) identify groups of components in language models that encode sentiment (Radford et al., 2017), part-of-speech tags (Blevins et al., 2018), and syntactic structure (Hewitt & Liang, 2019). Finally, mechanistic interpretability analyses (Wang et al., 2022; Nanda et al., 2023) uncover specific components that encode a model behavior of interest, e.g., "knowledge neurons" (Dai et al., 2021) and "induction heads" (Olsson et al., 2022). Broadly, these works leverage different tools to answer the question: *How do individual model components shape model behavior?*

In this work, we propose a new (and complementary) approach to studying this question. Our point of start is to rephrase the question, instead asking:

How do changes to model components collectively change individual model predictions?

We turn this rephrased question into a concrete framework called component modeling. In this framework, the goal is to build an interpretable *counterfactual estimator* of how a model's output would change in response to interventions made to its components. In the rest of the paper, we present a general approach to building such estimators, which turn out to be highly predictive in large-scale settings. Beyond shedding light on how model components collectively contribute to a given prediction, these estimators enable effective model editing, allowing us to design targeted interventions that induce a desired change in model behavior.

Roadmap and contributions. The main contribution of our work is a framework for decomposing model predictions in terms of model components, which we show has direct applications to model editing. Figure 1 summarizes these contributions. Specifically, in this paper we:

1. **Introduce the component modeling framework:** We formalize our goal of understanding how model components shape predictions through a framework called *component modeling* (Definition 1). The objective is to learn a counterfactual estimator, or *component model*, that predicts the effect of ablating a subset of components on a model prediction (Equation 1). Intuitively, this framework operationalizes the idea that if we can "understand" how model components shape a prediction, we should be able to estimate how the prediction would change if we were to ablate a set of components.
2. **Instantiate the framework via component attribution:** We focus our attention on a special "linear" case of component modeling called *component attribution*,

[1]MIT. Correspondence to: Harshay Shah <harshay@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

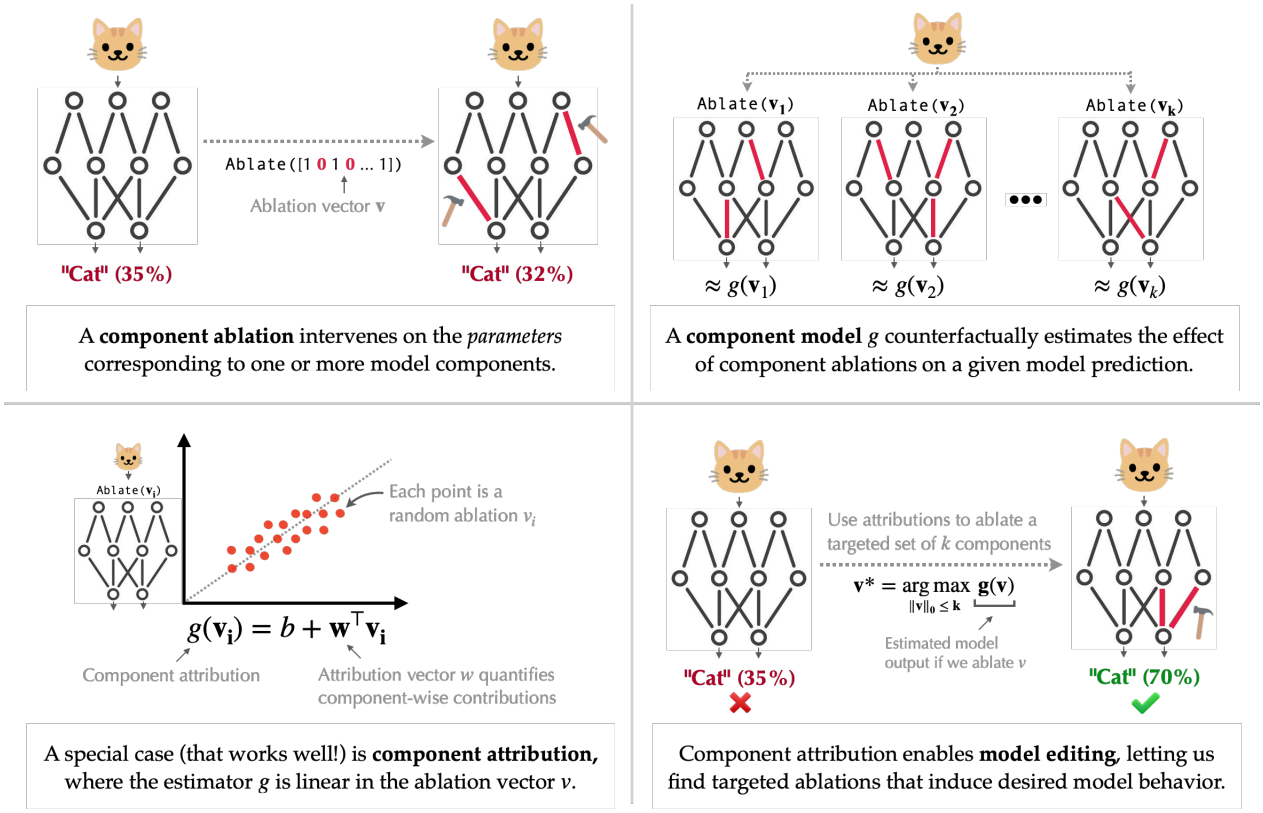

Figure 1: A summary of the component modeling framework.

where we assign a score to each component, and estimate the counterfactual effect of ablating a set of components as the sum of their corresponding scores (Definition 2). Component attributions surface the "contribution" of every component to a prediction while abstracting away the complexity of the model's computation.

3. **Propose an algorithm for efficient component attribution:** We develop COAR (component attribution via regression), a scalable way to estimate component attributions (Section 3). Through experiments on both image classifiers and language models, we show that COAR yields component attributions that can accurately predict how model predictions change in response to component-level ablations (Section 4).
4. **Edit model behavior via component attribution:** COAR attributions can directly enable edits to large-scale classifiers without additional training (Section 5). In Section 5, we outline COAR-EDIT, an editing method that uses COAR attributions as counterfactual estimators to design component ablations that change model behavior as targeted. Then, we apply COAR-EDIT to five editing tasks: fixing model errors (§ 5.1), boosting subpopulation robustness (§ 5.2), improving CLIP robustness to typographic attacks (§ 5.3), "forgetting" classes (§ D.1), and localizing backdoor attacks (§ D.2).

Paper organization. We begin by introducing the component modeling framework and its special case, component attribution, in Section 2. We then describe COAR, our component attribution estimator in Section 3, and showcase its effectiveness on large-scale models in Section 4. Finally, we apply COAR to model editing in Section 5.

2. Setup and Problem Statement

Consider a typical supervised learning setup. We have a set S of input-label pairs (or *examples*) $z_i = (x_i, y_i)$, and a trained model M that maps inputs x to predicted labels $M(x)$. We define the *model output function* $f_M(z) \in \mathbb{R}$ as any statistic that quantifies the correctness of model M on the example z e.g., the cross-entropy loss in a classification task, or the squared loss in a regression task.

In this work, we will think of the model M not as a black box, but instead as the output of a *computation graph* G_M (Bauer, 1974). Each parameterized node of this graph—which we call a *component*—is a function mapping its incoming edges to an outgoing edge. For example, a d-dimensional linear model M naturally admits a computation graph G_M with d components—one component $C_i(z) = w_i \cdot z$ for each parameter w_i—followed by a summation that combines the components into an output.

For more complex models, there are often multiple valid computation graphs G_M we could consider. For example, if M is a Transformer (Vaswani et al., 2017), the components might be multi-head attention blocks, individual attention heads, or even individual parameters. In general, the component set depends on the model architecture *and* the level of granularity that we wish to study.

Component modeling. Our goal in this work is to understand the behavior of the model M in terms of its components. By viewing the model M as a computation graph G_M over a set of components C, we can restate our goal:

Given a model M and example z, how do individual components $c \in C$ combine to yield the output $f_M(z)$?

Of course, there is a trivial answer to this question: the components $c \in C$ combine through the very computation graph used to define C. This answer is correct but not satisfying, as it does not get us closer to our conceptual goal of understanding model behavior in terms of components.

What we are truly after is a *simple, interpretable* function capturing how components in C impact $f_M(z)$. To make this more precise, we define the *component counterfactual function* $f_M(z, C')$ as

$$f_M(z, C') := \text{model output } f_M(z) \text{ on example } z \text{ \textit{after} ablating components } C' \subseteq C, \quad (1)$$

where "ablating" here corresponds to any intervention that overrides or patches the parameters corresponding to components $c \in C'$ (e.g., by setting them to zero (Olsson et al., 2022) or by adding random noise (Meng et al., 2022)).

Equation (1) allows us to operationalize our goal as a *counterfactual estimation* task. In this task, we want to estimate component counterfactuals $f_M(z, C')$ using a much simpler function, which we call a component model.

Definition 1 (Component modeling)**.** *Fix a model M with computation graph G_M, component set $C = \{c_1, \ldots, c_N\}$, and model output function f_M. For any subset of model components $C' \subseteq C$, let $\mathbf{0}_{C'}$ be the corresponding ablation vector of C', defined as a N-dimensional vector where*

$$(\mathbf{0}_{C'})_i = \begin{cases} 0 & \text{if } c_i \in C' \\ 1 & \text{otherwise.} \end{cases}$$

Given an example z, a <u>component model</u> for z is a function $g^{(z)} : \{0,1\}^N \to \mathbb{R}$ that maps ablation vectors of subsets C' to estimates of the counterfactual $f_M(z, C')$.

In other words, the high-level goal of component modeling is to build an estimator that can directly answer counterfactual questions like *"what would happen to my classifier's prediction on a given image if I ablated a specific set of components $C' \subseteq C$?"* without having to intervene on the computation graph G_M and ablate components in C'.

Component attribution. In this work, we consider a subcase of component modeling—which we call *component attribution*—where the function $g^{(z)}$ is *linear* in its input. That is, a component attribution for example z assigns a score $\boldsymbol{w}_i^{(z)}$ to each component $c_i \in C$, and predicts the effect of ablating $C' \subset C$ as the sum of the scores corresponding to components in $C \setminus C'$.

Definition 2 (Component attribution)**.** *Fix a model M with output function f_M and component set $C = \{c_1, \ldots, c_N\}$. A <u>component attribution</u> for example z is a set of parameters $\boldsymbol{\theta}^{(z)} := \{\boldsymbol{w}_1^{(z)}, \ldots, \boldsymbol{w}_N^{(z)}, b^{(z)}\}$ which parameterize a linear component model, i.e., a function $g^{(z)}$ such that*

$$f_M(z; C') \approx g^{(z)}(\mathbf{0}_{C'}) := \mathbf{0}_{C'}^\top \boldsymbol{w}^{(z)} + b^{(z)}$$

Component attribution satisfies our goal of finding a simple, interpretable account of how model components combine to form predictions. In particular, a component attribution for example z decomposes a model's output on z into the contributions $\boldsymbol{w}_i^{(z)}$ of each component c_i.

Remark 1 (Linearity and misspecification)**.** *ML models can comprise computation graphs with highly non-linear interactions among model components. For such models, it is a prior unclear why the effect of ablating (1) should be well-approximated by <u>linear</u> component attributions (2). Despite this evident misspecification, our results on vision and language models in Section 4 show that linear attributions <u>can</u> accurately predict component counterfactuals.*

3. Component attribution with COAR

In Section 2, we formalized our goal of understanding how models internally process examples into a counterfactual estimation task via *component modeling* (Definition 1), of which we study a special (linear) case called *component attribution* (Definition 2). Now, we estimate component attributions $\boldsymbol{\theta}^{(z)}$ by casting the task as a regression problem. Specifically, we describe COAR (<u>co</u>mponent <u>a</u>ttribution via <u>r</u>egression), a general component attribution method for models ranging from random forests to deep networks.

Approach. Consider a fixed model output $f_M(\cdot)$ of interest, and a computation graph G_M that encodes the model components C at the desired level of granularity. Additionally, we fix an *ablation method*, i.e., a procedure for "overriding" or patching any given subset $C' \subset C$ of the model components in the computation graph G_M.

Our method COAR takes in an example z and outputs a component attribution vector $\boldsymbol{\theta}^{(z)} \in \mathbb{R}^{|C|+1}$ (Definition 2) by casting the task of predicting component counterfactuals as a *supervised learning* problem in two steps:

1. **Construct a component dataset.** We construct a dataset $D^{(z)}$ of component counterfactuals for the example z. Each "datapoint" in $D^{(z)}$ consists of a component subset $C_i \subseteq C$ and its counterfactual $f_M(z, C_i)$ (see (1))—we compute the latter by simply ablating the components in C_i and evaluating the model on example z. In this work, we choose the component subsets C_i to be random α_{train}-fraction subsets of the component set C, for a ablation fraction hyperparameter $\alpha_{\text{train}} > 0$.[1] The output of this step is a *component dataset*

$$D^{(z)} = \{(C_1, f_M(z, C_1)), ..., (C_m,\ f_M(z, C_m))\} \quad (2)$$

where $C_i \sim \text{Uniform}(\{C' \subset C : |C'| = \alpha_{\text{train}}|C|\})$. We study the effect of varying the ablation fraction α_{train} on COAR in Appendix I.1.

2. **Fit a linear estimator.** We then use the dataset $D^{(z)}$ to fit component attribution $\boldsymbol{\theta}^{(z)}$ for each example z (see Definition 2). Specifically, for each example z, we minimize the squared loss between the component counterfactuals from Step 1 and their corresponding attribution-based predictions by solving a *linear regression* problem:

$$\boldsymbol{\theta}^{(z)} := \arg\min_{b \in \mathbb{R},\, \boldsymbol{w} \in \mathbb{R}^{|C|}} \sum_{D^{(z)}} (b + \mathbf{1}_{C_i}^\top \boldsymbol{w} - f_M(z, C_i))^2, \quad (3)$$

where again $\mathbf{0}_{C_i}$ is the ablation vector of C_i (Definition 1). Our component model is then

$$g^{(z)}(\mathbf{0}_{C'}) := \mathbf{0}_{C'}^\top \boldsymbol{w}^{(z)} + b^{(z)}. \quad (4)$$

We provide pseudocode for COAR in Appendix E.1. The resulting component attribution $\boldsymbol{\theta}^{(z)} := (\boldsymbol{w}^{(z)}, b^{(z)})$ is interpretable in that the coefficient $\boldsymbol{w}_j^{(z)}$ estimates how the output on example z would change if we were to ablate component c_j. We can thus view this coefficient as the (estimated) contribution of component c_j to the model output.

The above two-step approach is simple and highly scalable—we can construct the dataset $D^{(z)}$ with forward passes on the given model to compute component counterfactuals, and optimize the linear regression problem (eq. 3) with off-the-shelf GPU-based solvers—see Appendix E.4 for details. This enables us to apply COAR to large-scale models and datasets, as shown in the next section.

Instantiating COAR for classifiers. Our method COAR is *general* in that we can use it to study any machine learning model M that has a corresponding output function f_M and computation graph G_M. In this work, we primarily use COAR to analyze models trained on classification tasks. Although the computation graph G_M will vary based on the specific model architecture we are studying, across all models we use the standard *correct-class margin* (Ilyas et al., 2022) as the model output f_M, i.e.,

$$f_M(z) := \text{(logit for correct class)} - \text{(highest logit for incorrect class)}, \quad (5)$$

a quantity whose sign indicates the correctness of model M on the example z. We choose to ablate component subsets $C' \subset S$ by simply setting the parameters of the components in C' to zero (Wang et al., 2022; Olsson et al., 2022). We use COAR with alternative model output functions and ablation methods in Appendices I.3 and I.2 respectively.

Remark 2 (Ablation is not removal)**.** *As noted in prior work (Chan et al., 2022), ablation methods (e.g., setting weights or activations to zero) do not "remove" model components from the computation graph. Instead, such ablations shift the activations off-distribution in a systematic way—the goal of component attribution (definition 2) is to predict the change in model outputs induced by this shift. We use zero ablations as it is a common choice in the literature (Olsson et al., 2022; Wang et al., 2022). In Appendix I.2, we show that* COAR *can estimate component attributions with alternative ablation methods as well.*

4. Does COAR learn accurate attributions?

We now evaluate whether COAR-estimated attributions predict component counterfactuals (1) for deep networks trained on image classification and language modeling.

Datasets, models, and components. We apply COAR to compute component attributions in three different setups (see Appendix E.2 for details):

- **Setup A:** A ResNet-18 (He et al., 2015) trained on the CIFAR-10 dataset (Krizhevsky, 2009), with a computation graph G_A comprising $|C| = 2,306$ components. Specifically, each model component $c_i \in C$ corresponds to a convolutional filter in the model, and ablating a set of components $C' \subset C$ means setting all the weights in the corresponding filters to zero.
- **Setup B:** A ResNet-50 trained on the ImageNet dataset (Deng et al., 2009), with a computation graph G_B comprising $|C| = 22,720$ components. Again, each component here corresponds to a convolutional filter in one of the 49 convolution layers of the ResNet-50.
- **Setup C:** A Vision Transformer (ViT-B/16) (Dosovitskiy et al., 2021) trained on ImageNet, whose computation graph G_C comprises $82,944$ components. Each component here corresponds to a row of a weight matrix in one of 12 ViT transformer blocks, and ablating a set of components means setting the corresponding rows to zero.

[1] We opt for this random α-fraction sampling method for simplicity—it may be possible to make COAR more statistically efficient by choosing the subsets C_i more carefully.

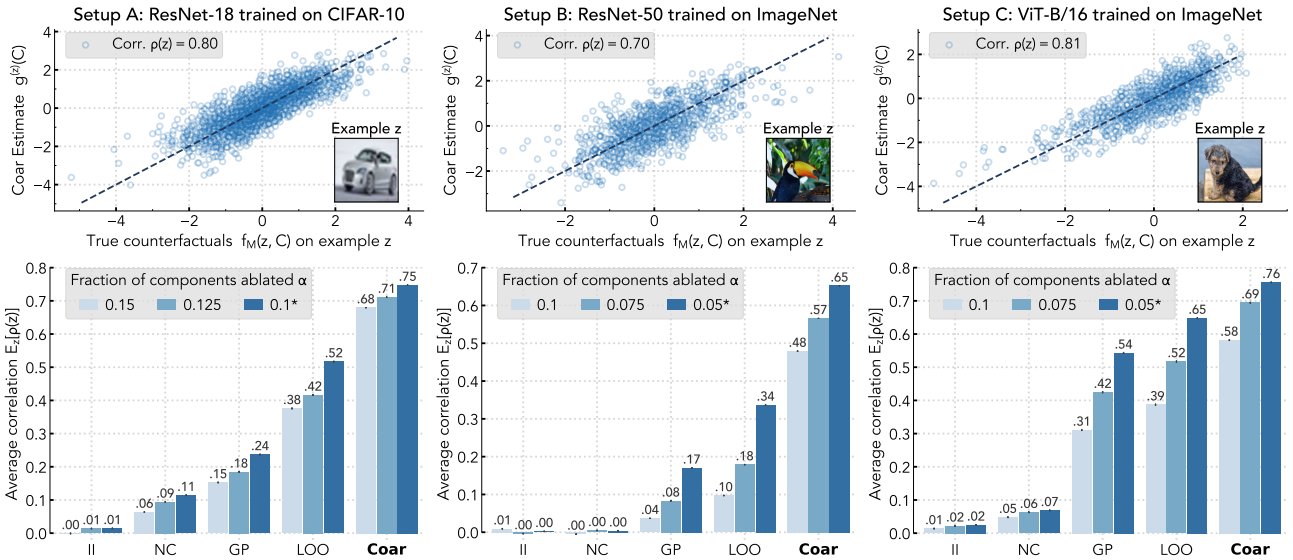

Figure 2: Evaluating COAR. We evaluate whether component attributions computed using our procedure COAR accurately predict component counterfactuals (1). We compare COAR to four baselines (described in Section 4) on three image classification setups (one per row). The subfigures on the left each focus on a single example z (visualized in the bottom-right corner of each plot), and show that for each setup, the ground-truth component counterfactuals $f_M(z, \cdot)$ (x-axis) and attribution-based estimates $g^{(z)}(\cdot)$ (y-axis) exhibit high correlation $\rho(z)$. On the right, we observe that COAR attributions exhibit high average correlation $\mathbb{E}_z[\rho(z)]$ over test examples, outperforming all baselines in each task and for all ablation fractions α_{test}. The asterisk (*) in each legend denotes α_{train}, the ablation fraction used to fit the component attributions.

Applying COAR. We use COAR to obtain component attributions (one for each test example) in each setup. Specifically, for a given model, we first construct a component dataset $D^{(z)}$ for each example z (as in Step 1 of Section 3) by randomly ablating α_{train} fraction of all components and evaluating the resulting margin (5) on z, where $\alpha_{\text{train}} = \{10\%, 5\%, 5\%\}$ for setup $\{\text{A}, \text{B}, \text{C}\}$ above. We repeat this m times, yielding a component dataset $D^{(z)}$ of size m for each example z—we use $m = \{50000, 100000, 200000\}$ for setup $\{\text{A}, \text{B}, \text{C}\}$ above. We then run linear regressions on these datasets (as in Step 2 of Section 3) to yield the component attributions. We defer details to Appendix E.4 and study the effect of the dataset size m and ablation fraction α_{train} on the attributions in Appendices G.4 and G.5.

Evaluation metric. We evaluate component attributions based on their ability to estimate *unseen* component counterfactuals (see (1)), i.e., the result of ablating component sets C' not collected in $D^{(z)}$. Specifically, we sample a new collection of k component subsets

$$D^{(z)} := \{C'_1, C'_2, \ldots, C_k\},$$
$$\text{where } C'_i \sim \text{Unif}(\{C' \subset C : |C'| = \alpha_{\text{test}}|C|\}).$$

where α_{test} is a hyperparameter. Setting $\alpha_{\text{test}} = \alpha_{\text{train}}$ ($\alpha_{\text{test}} \neq \alpha_{\text{train}}$) corresponds to evaluating the in-distribution (out-of-distribution) performance of COAR.

To quantify the predictiveness of component attributions, we use $D^{(z)}_{\text{test}}$ to measure the Pearson correlation between component counterfactuals $f_M(z, C'_i)$ and their corresponding attribution-based estimates $g^{(z)}(\mathbf{0}_{C'_i})$, i.e.,

$$\rho(z) := \rho_p\Big(\underbrace{\{f_M(z, C'_i)\}_{i=1}^k}_{\text{true counterfactuals}}, \underbrace{\{g^{(z)}(\mathbf{0}_{C'_i})\}_{i=1}^k}_{\text{attribution estimates}}\Big) \quad (6)$$

Baselines. We compare COAR attributions with four baselines, two adapted from related work and two natural baselines. We defer implementation details to Appendix E.3.

- **Adapted baselines (`NC`, `II`)**: We adapt **neuron conductance (`NC`)** (Dhamdhere et al., 2018) and **internal influence (`II`)** (Leino et al., 2018) to the our setting. Both methods use integrated gradients (Sundararajan et al., 2017) to compute importance scores for each component c_i. To compare these methods to COAR, we apply `NC` and `II` to model outputs on example z, and interpret the resulting scores as the attribution coefficients $w_i^{(z)}$.
- **Specialized baselines (`LOO`, `GP`):** We consider two more baselines. **Leave-one-out (`LOO`)** ablates individual components c_i and estimates the corresponding coefficient based on the effect of the ablation, setting $\boldsymbol{w}_i^{(z)} = f_M(z, \{c_i\}) - f_M(z, \emptyset)$. **Gradient-times-parameter (`GP`)**, which approximates the leave-one-out effect of each component using a first-order Taylor approximation, setting $\boldsymbol{w}_i^{(z)} = \nabla_{c_i} f_M(z, \emptyset) \cdot \delta_{c_i}$, where δ_{c_i} is the parameter-space change in c_i induced by the ablation method of choice.

4.1. Results

We use the setup described above to test whether COAR learns accurate component attributions for setups $\{\text{A}, \text{B}, \text{C}\}$. For each setup, we use COAR to estimate a component attribution for every example z in the corresponding test set. We then evaluate these attributions using the correlation metric $\rho(z)$ (6). Figure 2 depicts our results.

Example-level analysis. The first row of Figure 2 corresponds to an *individual test example* z from each task. For each example z, we ablate random component subsets $C' \subset C$ of size $\alpha_{\text{test}} \cdot |C|$ (for $\alpha_{\text{test}} = \alpha_{\text{train}}$) from the model computation graph and estimate the correlation $\rho(z)$ using Equation 6. Across all tasks, we observe that COAR learns accurate component attributions for the selected test examples. In Appendix G.6, we provide additional example-specific correlation plots for COAR and the baselines.

Aggregate analysis. The second row of Figure 2 shows the *average* correlation between the ground-truth counterfactuals and attribution-based estimates over test examples, i.e., $\mathbb{E}_z[\rho(z)]$. We also analyze the effect of subsampling fraction α_{test} on the average correlation, finding that:

(a) COAR outperforms all baselines by a large margin across datasets, models, and subsampling fractions α_{test}. For example, when randomly ablating $\alpha_{\text{test}} = 5\%$ of the components in the ImageNet ResNet50 (setup B), COAR estimators exhibit 66% correlation with the ground-truth component counterfactuals, whereas the estimates of the best-performing baseline, leave-one-out (LOO), exhibit 33% correlation.

(b) The correlation between ground-truth counterfactuals and their estimates for all attribution methods (including COAR) decays gracefully as α increases. For example, increasing α_{test} from 10% (equal to α_{train}) to 12.5% and 15% on CIFAR-10 (setup A) only decreases the average correlation of COAR estimators from 0.74 to 0.70 and 0.68 respectively. See Appendix I.1 for details.

Applying COAR to language models. Although we focus on vision models in this work, COAR is modality-agnostic. In Appendix F, we show that COAR, without any modification, yields predictive component attributions for language models. GPT-2 (Radford et al., 2019) evaluated on TinyStories (Eldan & Li, 2023) and Phi-2 (Javaheripi & Bubeck, 2023) evaluated on BoolQ (Clark et al., 2019).

Additional evaluation. In Appendix G, we show that COAR attributions are predictive for out-of-distribution inputs (§G.1), additional architectures (§G.2), additional tasks (§G.3), and different train-time ablation fractions (§G.4). We also show that COAR outperform baselines when trained with 2-5× fewer samples in Appendix G.5, and provide qualitative analysis in Appendix G.7.

5. Do COAR Attributions Enable Editing?

We now evaluate the practical utility of COAR by applying it to the problem of *model editing*. That is, we ask:

Is ablating model components identified via COAR *attributions an effective way to edit models?*

To answer this question, we first define model editing in our context and provide a simple method, COAR-EDIT, for translating component attributions into model edits. We apply this approach to edit model behavior on individual examples (§5.1), subpopulations (§5.2), and concepts (§5.3). Our findings show that COAR *directly* enables editing.

Problem setup. Consider a model M, a target distribution $\mathcal{D}_T$, and a reference distribution $\mathcal{D}_R$. A *model edit* on M is an intervention that aims to modify performance on the target examples $z \sim \mathcal{D}_T$ in a specific way, while leaving behavior on reference examples $z \sim \mathcal{D}_R$ unchanged. In its most general version, model edits can involve fine-tuning (Zhu et al., 2020), parameter updates (Bau et al., 2020a), or even architectural changes (Huang et al., 2023b).

Since our goal is to study model components, we restrict model edits to *ablation-based* interventions in this work. That is, we only consider interventions whose output can be expressed as component counterfactuals (see (1)). The goal of an editing method, then, is to identify components whose ablation changes performance on a given target set of examples S_T, without impacting behavior on a reference set of examples S_R. We turn this intuition into a precise definition of the ablation-based editing problem below.

Definition 3 (Editing models by ablating components)**.** *Consider a model M with computation graph G_M, component set $C = \{c_1, \ldots, c_N\}$, and model output function f_M. Let $\mathcal{D}_T$ and $\mathcal{D}_R$ denote target and reference distributions over examples, respectively. An (ϵ, δ)-effective model edit for $\mathcal{D}_R$ and $\mathcal{D}_T$ is an intervention that ablates a subset of components $C_{edit} \in 2^C$ such that*

$$\underbrace{\mathbb{E}_{\mathcal{D}_R}[|f_M(z, C_{edit}) - f_M(z, \emptyset)|]}_{\textit{Effect of edit on reference examples is small}} \leq \epsilon, \quad \textit{and}$$
$$\underbrace{\mathbb{E}_{\mathcal{D}_T}[|f_M(z, C_{edit}) - f_M(z, \emptyset)|]}_{\textit{Effect of edit on target examples is large}} \geq \delta, \tag{7}$$

where $f_M(z, C')$ denotes the model output function f_M evaluated on example z after ablating components C'.

As per Definition 3, every component subset $C' \subset 2^C$ defines a potential model edit. Effectively editing the model (as in (7)), however, requires identifying a subset C' that, when ablated, significantly changes outputs on the target distribution but not on the reference distribution.

An attribution-based approach to model editing. In this section, we show that an effective component attribution

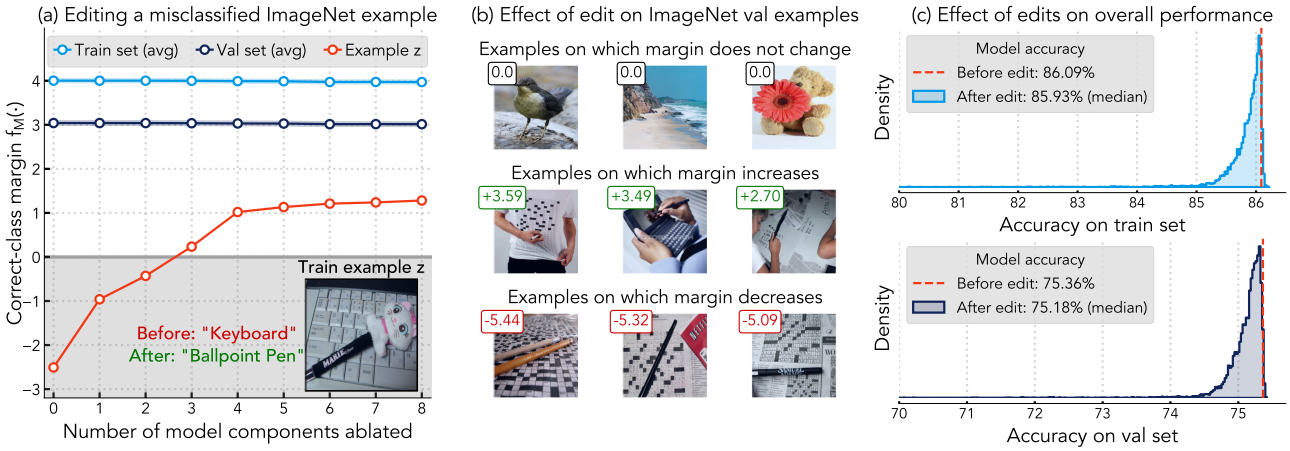

Figure 3: Editing individual model predictions with COAR-EDIT. We edit a ResNet50 model to correct a misclassified ImageNet example z shown on the left. Ablating a few components via COAR-EDIT (see (9)) increases the correct-class margin (5) on example z (red) without changing the average margin on the train set (light blue) or validation set (dark blue). In the middle, we observe that the examples on which model outputs change the least (top row) due to the edit are visually dissimilar to example z as well as examples on which model outputs change most positively (middle row) and negatively (bottom row). On the right, we find that individually performing model edits to correct every misclassified example in the validation set incurs a median accuracy drop of at most 0.2% on the train set (top row) and validation set (bottom row).

method (like COAR) can directly identify effective model edits. Key to this utility is a fundamental connection between component attribution and model editing. In particular, the former answers questions of the form, "*how would the model outputs change if we were to ablate a subset of components?*" while the latter inverts this question to ask "*which components, when ablated, would change model outputs in a specific way?*" By identifying the components that are most "important" to the desired model outputs, an attribution method can thus identify the right set of components to target via ablation-based editing (Definition 3).

To make this concrete, we propose COAR-EDIT, a simple three-step editing approach based on COAR attributions. Specifically, given a model M with a set of model components C, a set of target examples S_T sampled from $\mathcal{D}_T$, and a set of reference examples S_R sampled from $\mathcal{D}_R$, COAR-EDIT identifies a model edit (7) in three steps:

1. Estimate COAR attributions $\boldsymbol{\theta}^{(z)} := (\boldsymbol{w}^{(z)}, b^{(z)})$ where $\boldsymbol{w}^{(z)} \in \mathbb{R}^{|C|}$ and $b^{(z)} \in \mathbb{R}$ for every target and reference example $z \in S_T \cup S_R$.

2. For each component $c_i \in C$, use a t-test in order to quantify the "importance" of component c_i to set of target examples S_T relative to reference examples S_R:

$$\tau(c_i) := \frac{\mu(S_T) - \mu(S_R)}{\sqrt{\frac{\sigma^2(S_T)}{|S_T|} + \frac{\sigma^2(S_R)}{|S_R|}}}, \tag{8}$$

$$\text{where} \begin{cases} \mu(S) = \frac{1}{|S|} \sum_{z \in S} w_i^{(z)} \\ \sigma^2(S) = \frac{1}{|S|} \sum_{z \in S} (w_i^{(z)} - \mu(S))^2. \end{cases}$$

3. Increase the model outputs on target examples, ablate a set of components C_{edit} comprising the k most negative scores τ_i, i.e., set

$$C_{\text{edit}} = \arg\text{bottom-}k(\{\tau(c_i) : c_i \in C\}), \tag{9}$$

where the number of ablated components k is a hyperparameter that one can tune by cross-validation. Similarly, if the goal is to *decrease* model outputs on S_T, we replace bottom-k with top-k in Equation 9.

To make sense of the approach above, note that for every component c_i and a set of examples S, $\mu(S)$ in Equation 8 leverages attributions to directly estimate $\mu_i(S)$, the average effect of ablating c_i on model predictions for samples in set S. Similarly, the term $\sigma_i^2(S)$ captures the variation (across examples) of this effect. As a result, the score $\tau(c_i)$ in Equation 8 exactly corresponds to the two-sample t-test statistic, with a null hypothesis that the component c_i has an equal average effect on the target distribution $\mathcal{D}_T$ and the reference distribution $\mathcal{D}_R$. We then use these scores $\{\tau(c_i) : c_i \in C\}$ in Equation 9 to identify components that, if ablated, would change the target outputs the most relative to the change in the reference outputs.

5.1. Editing individual model predictions

In this section, we test whether COAR-EDIT can modify individual predictions of an ImageNet ResNet50 classifier (Setup B in Section 4) without impacting its overall performance. Specifically, we study the case where the target distribution $\mathcal{D}_T$ is a singleton example on which we want

to improve performance. An effective model edit (definition 3) here would increase the model margin (5) on z to be greater than zero without degrading overall performance.

Results. We apply COAR-EDIT to edit individual misclassified examples z, setting $S_T = \{z\}$ and S_R to be a small set of random samples from the ImageNet dataset. We present our findings in Figure 3. Figure 3a illustrates a single such edit, where we correct the model's prediction on a specific ImageNet example from "keyboard" to "ballpoint pen" by ablating $k = 3$ components (0.01% of all components). Specifically, increasing the number of ablated components k consistently improves the correct-class margin on target example z (red) without changing the average margin over the training set (light blue) or validation set (dark blue). Figure 3b then visualizes (again, for the specific example being edited in Figure 3a) the examples on which model outputs changes most (and least) drastically. Finally, Figure 3c shows that we can individually fix *every* misclassification in the ImageNet validation set while incurring a median accuracy drop of 0.2% on the training set (top row) and validation set (bottom row). We defer additional details and results to Appendix H.1.

5.2. Improving subpopulation robustness

Machine learning models often latch onto spurious correlations in the training dataset (Geirhos et al., 2019; Shah et al., 2020; Hermann et al., 2023), resulting in subpar performance on subpopulations where these correlations do not hold (Buolamwini & Gebru, 2018; Oakden-Rayner et al., 2020). In this section, we test whether our editing approach can boost performance on such underperforming subpopulations without degrading overall performance.

In particular, we evaluate COAR-EDIT on two benchmarks —Waterbirds (Sagawa et al., 2020) and CelebA (Liu et al., 2015)—where models fare poorly on subpopulations that are underrepresented in the training data. On both datasets, our goal is to improve a given model's *worst-subpopulation accuracy*—we defer details to Appendix H.3.

Results. On both datasets, COAR successfully identifies component subsets that correspond to effective model edits. On Waterbirds (Figure 4a), ablating 0.9% of all components improves worst-subpopulation accuracy from 64% to 83% (red) without degrading its accuracy uniformly averaged over examples and subpopulations. On CelebA, Figure 4b shows that zeroing out 26 of $22,720$ components improves worst-subpopulation accuracy from 47% to 85% and average-subpopulation accuracy from 84% to 90% while only incurring a 5% drop in test set accuracy.

Before continuing, we make two observations. First, COAR-EDIT is *sample-efficient*—it does not require subpopulation-level annotations for the training set; only

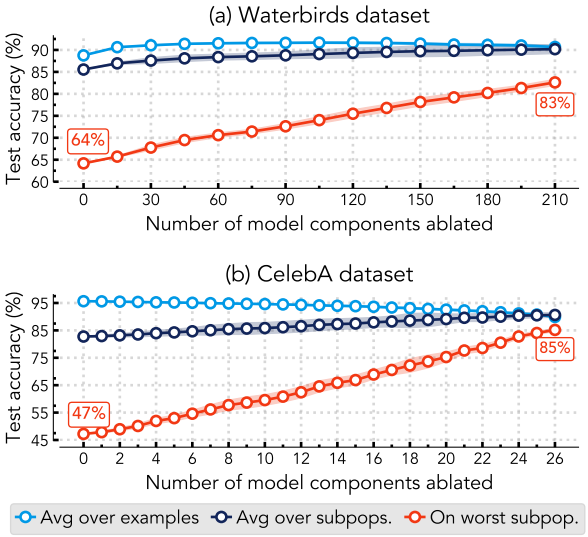

Figure 4: Improving subpopulation robustness with COAR-EDIT. We edit pre-trained ResNet-50 models to improve their worst-subpopulation accuracy on Waterbirds (Sagawa et al., 2020) and CelebA (Liu et al., 2015). Before editing, Waterbirds and CelebA models attain 87% and 96% test accuracy but only 64% and 47% accuracy on their worst-performing subpopulations, respectively. On the left, applying COAR-EDIT by ablating 210 of $22,720$ components in the Waterbirds model increases its worst-subpopulation accuracy from 64% to 83% without degrading its accuracy on examples (light blue) and subpopulations (dark blue). On the right, editing the CelebA model by ablating just 26 components improves worst-subpopulation accuracy from 47% to 85%.

20 random examples from each subpopulation suffice. Second, our results show that simply ablating a few components from models trained via "standard" empirical risk minimization (ERM) can lead to worst-subpopulation accuracy improvements comparable to gains from specialized methods (e.g., via robust optimization (Sagawa et al., 2020), dataset selection (Idrissi et al., 2022))

5.3. Improving robustness to typographic attacks

Zero-shot CLIP classifiers (Radford et al., 2021) are vulnerable to typographic attacks (Goh et al., 2021) that simply overlay text on images to induce misclassifications. We evaluate whether COAR-EDIT can improve the zero-shot robustness of a CLIP ViT-B/16 classifier using data (Materzyńska et al., 2022) comprising 180 images with and without multiple typographic attacks. Specifically, we use COAR-EDIT to identify component subset that, when ablated, fix the misclassifications induced by synthetic attacks without impacting predictions on images without attacks. We defer details to Appendix H.5.

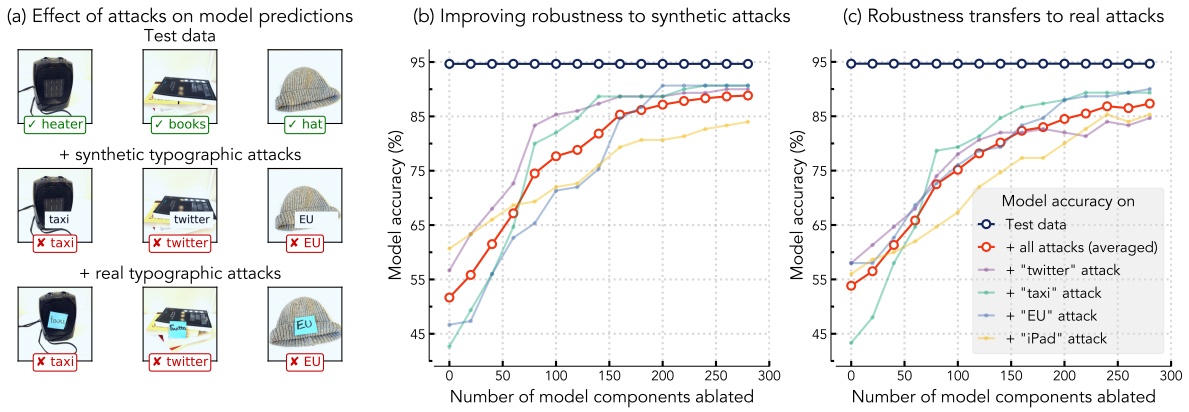

Figure 5: Improving robustness to typographic attacks with COAR-EDIT. We edit a zero-shot CLIP ViT-B/16 classifier to improve its robustness to typographic attacks (Goh et al., 2021). On the left, we find that predictions on images of household objects (top row) can be manipulated to "taxi", "twitter", or "EU" via synthetic (middle row) and real (last row) attacks. In the center panel, we use COAR-EDIT to identify components that, when ablated, improve average accuracy on examples with synthetic attacks (red) from 51% to 89% while maintaining accuracy on examples without attacks (blue). On the right, we find that the edit transfers robustness to real attacks too, improving accuracy from 54% to 86% on average.

Results. Figure 5 summarizes our findings. In Figure 5a, we show that the predictions of a zero-shot CLIP ViT-B/16 classifier on images of household objects (top row) can be manipulated to "taxi", "twitter", or "EU" via synthetic (middle row) or real (last row) typographic attacks. More quantitatively, we find that the zero-shot accuracy on images with synthetic and real typographic attacks drops from 95% to 51% and 54%, respectively. Figure 5b shows that ablating a subset of 300 components (0.4%) identified via COAR-EDIT improves the accuracy on held-out images with synthetic typographic attacks from 51% to 89% on average (red), without impacting accuracy on images without attacks (dark blue). Furthermore, in Figure 5c, we find that our edit transfers robustness to real typographic attacks as well, improving accuracy on held-out images from 54% to 86% on average. Similar to previous experiments, our approach is sample-efficient in that it only requires 15 pairs of target and reference examples with and without synthetic attacks to identify the edit described above.

Additional experiments. We also apply COAR-EDIT to two additional settings in Appendix D: selectively "forgetting" a class (§D.1) and localizing backdoor attacks (§D.2).

To summarize, simply ablating targeted subsets of components identified via COAR-EDIT can induce specific model behavior without requiring additional training. More broadly, our findings highlight how accurate component attribution *alone* can directly inform model editing.

6. Related work

In addition to the bottom-up approaches for studying model components discussed in Section 1, component modeling connects to prior work in interpretability and model editing.

Localizing model behavior. For one, COAR connects to a line of work in mechanistic interpretability on identifying "circuits" or "subnetworks" within neural networks that are responsible for specific behaviors (Olah et al., 2020b; Cao et al., 2021; Vig et al., 2020; Olsson et al., 2022; Wang et al., 2022), and in particular finding these subnetworks in an automated manner (Conmy et al., 2023; Bayazit et al., 2023). While recent work shows that these methods can be sensitive to design choices (Zhang & Nanda, 2023) and lack actionable insights for model editing (Hase et al., 2023), we show that COAR is agnostic to the choice of ablation method and directly enables model editing (Section 5).

Editing model behavior. Another line of work is on *model editing*, where one tries to induce or suppress specific behaviors via targeted changes to model parameters (Bau et al., 2020a; Santurkar et al., 2021; Ilharco et al., 2022; Belrose et al., 2023). In Section 5, we show that COAR can enable editing by simply zeroing out specific components.

We discuss additional related work, limitations, and extensions to COAR in Appendices A, B, and C respectively.

7. Conclusion

We introduce *component modeling*, a framework for decomposing predictions in terms of model components. We specifically focus on *component attribution*, a special case of this framework, where the goal is to predict the counterfactual impact of every component on a given prediction. Our proposed method for this task, COAR, yields predictive attributions for large-scale vision and language models. We then demonstrate how COAR attributions can directly enable effective model editing without requiring additional training or data collection.

Impact Statement

Our work seeks to improve our understanding of black-box machine learning models. We do not foresee any significant societal consequences in the short term. However, in the long term, we believe that a concrete understanding of deep learning phenomena is essential to develop reliable deep learning systems for practical applications that have societal impact.

Acknowledgements

The authors would like to thank Benjamin Cohen-Wang, Logan Engstrom, Alaa Khaddaj, Kristian Georgiev, and Sung-Min Park for helpful discussions and comments on an earlier draft of this manuscript.

Work supported in part by the NSF grant DMS-2134108. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0015.

References

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Antverg, O. and Belinkov, Y. On the pitfalls of analyzing individual neurons in language models. *arXiv preprint arXiv:2110.07483*, 2021.

Basu, S., Zhao, N., Morariu, V., Feizi, S., and Manjunatha, V. Localizing and editing knowledge in text-to-image generative models. *arXiv preprint arXiv:2310.13730*, 2023.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Bau, D., Liu, S., Wang, T., Zhu, J.-Y., and Torralba, A. Rewriting a deep generative model. In *European Conference on Computer Vision (ECCV)*, 2020a.

Bau, D., Zhu, J.-Y., Strobelt, H., Lapedriza, A., Zhou, B., and Torralba, A. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences (PNAS)*, 2020b.

Bauer, F. L. Computational graphs and rounding error. In *SIAM Journal on Numerical Analysis*, volume 11, pp. 87–96. SIAM, 1974.

Bayazit, D., Foroutan, N., Chen, Z., Weiss, G., and Bosselut, A. Discovering knowledge-critical subnetworks in pretrained language models. *arXiv preprint arXiv:2310.03084*, 2023.

Beery, S., Van Horn, G., and Perona, P. Recognition in terra incognita. In *European Conference on Computer Vision (ECCV)*, 2018.

Belrose, N., Schneider-Joseph, D., Ravfogel, S., Cotterell, R., Raff, E., and Biderman, S. Leace: Perfect linear concept erasure in closed form. *arXiv preprint arXiv:2306.03819*, 2023.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2023.

Blevins, T., Levy, O., and Zettlemoyer, L. Deep rnns encode soft hierarchical syntax. *arXiv preprint arXiv:1805.04218*, 2018.

Bolukbasi, T., Pearce, A., Yuan, A., Coenen, A., Reif, E., Viégas, F., and Wattenberg, M. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

Brown, D., Godfrey, C., Nizinski, C. A., Tu, J., and Kvinge, H. Robustness of edited neural networks. *ArXiv*, abs/2303.00046, 2023.

Buolamwini, J. and Gebru, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency (FAccT)*, 2018.

Cammarata, N., Goh, G., Carter, S., Schubert, L., Petrov, M., and Olah, C. Curve detectors. *Distill*, 5(6):e00024–003, 2020.

Cao, S., Sanh, V., and Rush, A. M. Low-complexity probing via finding subnetworks. *arXiv preprint arXiv:2104.03514*, 2021.

Casper, S., Rauker, T., Ho, A., and Hadfield-Menell, D. Sok: Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *First IEEE Conference on Secure and Trustworthy Machine Learning*, 2022.

Chan, L., Garriga-Alonso, A., Goldowsky-Dill, N., Greenblatt, R., Nitishinskaya, J., Radhakrishnan, A., Shlegeris, B., and Thomas, N. Causal scrubbing: A method for rigorously testing interpretability hypotheses. 2022.

Chang, T.-Y., Thomason, J., and Jia, R. Do localization methods actually localize memorized data in llms? *arXiv preprint arXiv:2311.09060*, 2023.

Chen, H., Yang, J., Vondrick, C., and Mao, C. Interpreting and controlling vision foundation models via text explanations. *arXiv preprint arXiv:2310.10591*, 2023.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Cohen, R., Biran, E., Yoran, O., Globerson, A., and Geva, M. Evaluating the ripple effects of knowledge editing in language models. *ArXiv*, abs/2307.12976, 2023.

Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023.

Dai, D., Dong, L., Hao, Y., Sui, Z., Chang, B., and Wei, F. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Dalvi, F., Durrani, N., Sajjad, H., Belinkov, Y., Bau, A., and Glass, J. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6309–6317, 2019.

De Cao, N., Aziz, W., and Titov, I. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021a.

De Cao, N., Schmid, L., Hupkes, D., and Titov, I. Sparse interventions in language models with differentiable masking. *arXiv preprint arXiv:2112.06837*, 2021b.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

Dhamdhere, K., Sundararajan, M., and Yan, Q. How important is a neuron? *arXiv preprint arXiv:1805.12233*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Durrani, N., Sajjad, H., Dalvi, F., and Belinkov, Y. Analyzing individual neurons in pre-trained language models. *arXiv preprint arXiv:2010.02695*, 2020.

Eldan, R. and Li, Y. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Feldman, V. and Zhang, C. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 2881–2891, 2020.

Finlayson, M., Mueller, A., Gehrmann, S., Shieber, S., Linzen, T., and Belinkov, Y. Causal analysis of syntactic agreement mechanisms in neural language models. *arXiv preprint arXiv:2106.06087*, 2021.

Friedman, D., Lampinen, A., Dixon, L., Chen, D., and Ghandeharioun, A. Interpretability illusions in the generalization of simplified models. *arXiv preprint arXiv:2312.03656*, 2023.

Gandelsman, Y., Efros, A. A., and Steinhardt, J. Interpreting clip's image representation via text-based decomposition. *arXiv preprint arXiv:2310.05916*, 2023.

Gandikota, R., Materzynska, J., Fiotto-Kaufman, J., and Bau, D. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Geiger, A., Lu, H., Icard, T., and Potts, C. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586, 2021.

Geiger, A., Potts, C., and Icard, T. Causal abstraction for faithful model interpretation. *arXiv preprint arXiv:2301.04709*, 2023.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. In *Nature Machine Intelligence*, 2020.

Geirhos, R., Zimmermann, R. S., Bilodeau, B., Brendel, W., and Kim, B. Don't trust your eyes: on the (un) reliability of feature visualizations. *arXiv preprint arXiv:2306.04719*, 2023.

Georgiev, K., Vendrow, J., Salman, H., Park, S. M., and Madry, A. The journey, not the destination: How data guides diffusion models. *arXiv preprint arXiv:2312.06205*, 2023.

Ghiasi, A., Kazemi, H., Borgnia, E., Reich, S., Shu, M., Goldblum, M., Wilson, A. G., and Goldstein, T. What do vision transformers learn? a visual exploration. *arXiv preprint arXiv:2212.06727*, 2022.

Ghorbani, A. and Zou, J. Y. Neuron shapley: Discovering the responsible neurons. *Advances in neural information processing systems*, 33:5922–5932, 2020.

Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., Radford, A., and Olah, C. Multimodal neurons in artificial neural networks. *Distill*, 2021.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Hase, P., Bansal, M., Kim, B., and Ghandeharioun, A. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Hendrycks, D. and Dietterich, T. G. Benchmarking neural network robustness to common corruptions and surface variations. In *International Conference on Learning Representations (ICLR)*, 2019.

Hermann, K. L., Mobahi, H., Fel, T., and Mozer, M. C. On the foundations of shortcut learning. *arXiv preprint arXiv:2310.16228*, 2023.

Hernandez, E., Schwettmann, S., Bau, D., Bagashvili, T., Torralba, A., and Andreas, J. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*, 2021.

Hewitt, J. and Liang, P. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368*, 2019.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*, 2018.

Huang, J., Geiger, A., D'Oosterlinck, K., Wu, Z., and Potts, C. Rigorously assessing natural language explanations of neurons. *arXiv preprint arXiv:2309.10312*, 2023a.

Huang, Z., Shen, Y., Zhang, X., Zhou, J., Rong, W., and Xiong, Z. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*, 2023b.

Idrissi, B. Y., Arjovsky, M., Pezeshki, M., and Lopez-Paz, D. Simple data balancing achieves competitive worst-group-accuracy. In *Conference on Causal Learning and Reasoning*, pp. 336–351. PMLR, 2022.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. Datamodels: Predicting predictions from training data. In *International Conference on Machine Learning (ICML)*, 2022.

Javaheripi, M. and Bubeck, S. Phi-2: The surprising power of small language models, Dec 2023.

Johnson, A. E., Pollard, T. J., Greenbaum, N. R., Lungren, M. P., Deng, C.-y., Peng, Y., Lu, Z., Mark, R. G., Berkowitz, S. J., and Horng, S. Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042*, 2019.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Krizhevsky, A. Learning multiple layers of features from tiny images. In *Technical report*, 2009.

Lakretz, Y., Kruszewski, G., Desbordes, T., Hupkes, D., Dehaene, S., and Baroni, M. The emergence of number and syntax units in lstm language models. *arXiv preprint arXiv:1903.07435*, 2019.

Leclerc, G., Ilyas, A., Engstrom, L., Park, S. M., Salman, H., and Madry, A. ffcv. https://github.com/libffcv/ffcv/, 2022.

Leino, K., Sen, S., Datta, A., Fredrikson, M., and Li, L. Influence-directed explanations for deep convolutional networks. In *2018 IEEE international test conference (ITC)*, pp. 1–8. IEEE, 2018.

Li, M., Davies, X., and Nadeau, M. Circuit breaking: Removing model behaviors with targeted ablation. 2023a.

Li, Y., Yang, X., Sun, P., Qi, H., and Lyu, S. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3207–3216, 2020.

Li, Y., Bubeck, S., Eldan, R., Del Giorno, A., Gunasekar, S., and Lee, Y. T. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023b.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.

Maini, P., Mozer, M. C., Sedghi, H., Lipton, Z. C., Kolter, J. Z., and Zhang, C. Can neural network memorization be localized? In *International Conference on Machine Learning*, 2023.

Materzyńska, J., Torralba, A., and Bau, D. Disentangling visual and written concepts in clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16410–16419, 2022.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36, 2022.

Mitchell, E., Lin, C., Bosselut, A., Finn, C., and Manning, C. D. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Mu, J. and Andreas, J. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

Nanda, N. Attribution patching: Activation patching at industrial scale. 2023. *URL https://www. neelnanda. io/mechanistic-interpretability/attribution-patching*, 2023.

Nanda, N., Chan, L., Liberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Oakden-Rayner, L., Dunnmon, J., Carneiro, G., and Ré, C. Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. In *Proceedings of the ACM conference on health, inference, and learning*, 2020.

Oikarinen, T. and Weng, T.-W. Clip-dissect: Automatic description of neuron representations in deep vision networks. *arXiv preprint arXiv:2204.10965*, 2022.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. The building blocks of interpretability. In *Distill*, 2018.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. An overview of early vision in inceptionv1. *Distill*, 2020a. doi: 10.23915/distill.00024.002. https://distill.pub/2020/circuits/early-vision.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020b. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Palit, V., Pandey, R., Arora, A., and Liang, P. P. Towards vision-language mechanistic interpretability: A causal tracing tool for blip. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2856–2861, 2023.

Panigrahi, A., Saunshi, N., Zhao, H., and Arora, S. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*, 2023.

Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. Trak: Attributing model behavior at scale. In *Arxiv preprint arXiv:2303.14186*, 2023.

Radford, A., Jozefowicz, R., and Sutskever, I. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *arXiv preprint arXiv:2103.00020*, 2021.

Ravfogel, S., Twiton, M., Goldberg, Y., and Cotterell, R. D. Linear adversarial concept erasure. In *International Conference on Machine Learning*, pp. 18400–18421. PMLR, 2022.

Ribeiro, M. T., Singh, S., and Guestrin, C. " why should i trust you?" explaining the predictions of any classifier. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

Rosenfeld, E. and Risteski, A. Outliers with opposing signals have an outsized effect on neural network optimization. *arXiv preprint arXiv:2311.04163*, 2023.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2020.

Santurkar, S., Tsipras, D., Elango, M., Bau, D., Torralba, A., and Madry, A. Editing a classifier by rewriting its prediction rules. In *Preprint*, 2021.

Shah, H., Tamuly, K., Raghunathan, A., Jain, P., and Netrapalli, P. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33: 9573–9585, 2020.

Shah, H., Jain, P., and Netrapalli, P. Do input gradients highlight discriminative features? *Advances in Neural Information Processing Systems*, 34, 2021.

Shah, H., Park, S. M., Ilyas, A., and Madry, A. Modeldiff: A framework for comparing learning algorithms. In *International Conference on Machine Learning*, pp. 30646–30688. PMLR, 2023.

Sheng, E., Chang, K.-W., Natarajan, P., and Peng, N. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*, 2019.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Stolfo, A., Belinkov, Y., and Sachan, M. Understanding arithmetic reasoning in language models using causal mediation analysis. *arXiv preprint arXiv:2305.15054*, 2023.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Tigges, C., Hollinsworth, O. J., Geiger, A., and Nanda, N. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Vendrow, J., Jain, S., Engstrom, L., and Madry, A. Dataset interfaces: Diagnosing model failures using controllable counterfactual generation. *arXiv preprint arXiv:2302.07865*, 2023.

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., and Shieber, S. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33: 12388–12401, 2020.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset. 2011.

Wang, H., Ge, S., Xing, E. P., and Lipton, Z. C. Learning robust global representations by penalizing local predictive power. *Neural Information Processing Systems (NeurIPS)*, 2019.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. URL https://arxiv.org/abs/2211.00593.

Wang, Z., Yang, E., Shen, L., and Huang, H. A comprehensive survey of forgetting in deep learning beyond continual learning. *arXiv preprint arXiv:2307.09218*, 2023.

Wen, K., Li, Y., Liu, B., and Risteski, A. Transformers are uninterpretable with myopic methods: a case study with bounded dyck grammars. *arXiv preprint arXiv:2312.01429*, 2023.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhang, F. and Nanda, N. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

Zheng, X., Pang, T., Du, C., Jiang, J., and Lin, M. Intriguing properties of data attribution on diffusion models. *arXiv preprint arXiv:2311.00500*, 2023.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. Places: A 10 million image database for scene recognition. In *IEEE transactions on pattern analysis and machine intelligence*, 2017.

Zhou, B., Sun, Y., Bau, D., and Torralba, A. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*, 2018.

Zhu, C., Rawat, A. S., Zaheer, M., Bhojanapalli, S., Li, D., Yu, F., and Kumar, S. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*, 2020.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Appendices

A Related work **18**

B Discussion **19**

C Future work **20**

D Additional COAR-EDIT experiments **21**

D.1 "Forgetting" a class . . . 21

D.2 Mitigating a backdoor attack . . . 22

E Evaluation setup **23**

E.1 Pseudocode . . . 23

E.2 Datasets and models. . . . 23

E.3 Baselines . . . 24

E.4 Implementation details . . . 25

F Applying COAR to language models **26**

F.1 Evaluating GPT-2 on the TinyStories dataset . . . 26

F.2 Evaluating Phi-2 on the BoolQ dataset . . . 26

G Additional evaluation of COAR **28**

G.1 Evaluating COAR on additional datasets . . . 28

G.2 Evaluating COAR on additional model architectures . . . 28

G.3 Evaluating COAR on additional tasks . . . 28

G.4 Comparing COAR attributions estimated with different ablation fractions . . . 28

G.5 Comparing COAR attributions estimated with different sample sizes . . . 29

G.6 Analyzing COAR attributions at the example level . . . 29

G.7 Qualitatively analyzing COAR attributions . . . 29

H Additional evaluation of COAR-EDIT **38**

H.1 Editing individual predictions . . . 38

H.2 Forgetting a class . . . 38

H.3 Improving subpopulation robustness. . . . 38

H.4 Mitigating backdoor attacks. . . . 39

H.5 Improving robustness to typographic attacks. . . . 39

I Analyzing design choices in COAR **46**

I.1 Effect of ablation fraction 46
I.2 Effect of ablation method 46
I.3 Effect of model output function 47

A. Related work

Our work relates to several lines of work in machine learning interpretability, which we categorize into works that localize model behavior, works that interpret specific model components, and works that perform model editing.

Localizing model behavior. One line of work (within the field of *mechanistic* interpretability), attempts to localize specific capabilities or behaviors of neural networks (especially language models) to specific "subnetworks" or "circuits" (Olah et al., 2020b). For example, prior work has used a variety of methods to localize gender bias (Vig et al., 2020); specifical factual associations (Meng et al., 2022); and other behavior (Wang et al., 2022; Li et al., 2023a; Goldowsky-Dill et al., 2023; Tigges et al., 2023) within model parameters.

More recent work has tried to automate this localization process, using techniques based on fine-tuning (Panigrahi et al., 2023), activation patching (Conmy et al., 2023; Goldowsky-Dill et al., 2023), or differentiable masking (Bayazit et al., 2023; Cao et al., 2021; De Cao et al., 2021b; Chang et al., 2023). These techniques (or variants thereof) have been subsequently used to localize properties such as arithmetic reasoning (Stolfo et al., 2023), syntactic agreement (Finlayson et al., 2021), visual question answering (Palit et al., 2023), and visual attributes in diffusion models (Basu et al., 2023). Other work has also developed methods (Dhamdhere et al., 2018; Leino et al., 2018; Nanda, 2023; Ghorbani & Zou, 2020) to attribute model behavior to specific components.

Recently, however, Zhang & Nanda (2023) showed that the design choices underlying many automated localization methods (e.g., the way they ablate components) can drastically change their results. Furthermore, Hase et al. (2023) show that localizing factual associations does not directly inform how to erase or amplify these associations via model editing. In contrast, COAR can (a) adapt to any reasonable choice of ablation method (Remark 2) and (b) yield actionable insights for model editing (Section 5).

Editing model behavior. Another related line of work focuses on *model editing*, the goal of which is to make small, targeted changes to model representations in order to induce or suppress a specific behavior. Model editing methods include "hypernetworks" (De Cao et al., 2021a; Mitchell et al., 2021), rank-one updates to model parameters (Bau et al., 2020a; Santurkar et al., 2021; Meng et al., 2022), constrained fine-tuning (Zhu et al., 2020), and weight interpolation (Ilharco et al., 2022; Zou et al., 2023), among other methods. Recent work has also studied erasing concepts and suppressing spurious correlations from models using layer-wise linear probing (Belrose et al., 2023; Ravfogel et al., 2022), CLIP-specific text-based methods (Gandelsman et al., 2023; Chen et al., 2023), and fine-tuning variants (Gandikota et al., 2023; Kirichenko et al., 2022). In this work, we treat model editing as an application, and show how attributions can enable model edits that modify individual model predictions (§5.1,§D.1), improve subpopulation-level robustness (§5.2) and suppress spurious concepts (§D.2, §5.3).

Interpreting specific model components. Instead of starting with a functionality and trying to localize it to specific components, another line of work introduces methods for studying the functionality of individual model components. Such methods include, feature visualization (Zeiler & Fergus, 2014; Ghiasi et al., 2022; Olah et al., 2020a), activation maps (Bau et al., 2017; Mu & Andreas, 2020), ablations (Zhou et al., 2018), saliency maps (Olah et al., 2018), probing (Dalvi et al., 2019; Durrani et al., 2020), and natural language descriptions (Hernandez et al., 2021; Oikarinen & Weng, 2022; Bills et al., 2023). Subsequent analyses use these methods to identify and ascribe meaning to specific model components by labeling them as, e.g., "curve detectors" (Cammarata et al., 2020), "knowledge neurons" (Dai et al., 2021), "multimodal neurons" (Goh et al., 2021), and "syntax units" (Lakretz et al., 2019) to name a few. Recently, however, the reliability and robustness of these methods has been called into question (Bolukbasi et al., 2021; Antverg & Belinkov, 2021; Hewitt & Liang, 2019; Huang et al., 2023a; Geirhos et al., 2023; Hooker et al., 2018; Shah et al., 2021). Here, our goal is not to interpret specific model components, but rather to study how different components jointly influence model predictions through the lens of component modeling (Definition 1).

Understanding machine learning models by proxy. Finally, our work connects to a line of research that aims to understand machine learning models by constructing *interpretable proxies*. For example, certain feature attribution methods like LIME (Ribeiro et al., 2016) approximate a given ML model with a linear model in input space. Similarly, a line of work on datamodeling (Ilyas et al., 2022; Park et al., 2023) approximates a given learning algorithm by a linear model in "dataset space." More generally, one can view a component attribution (or in fact, any component model) as a *causal abstraction* (Geiger et al., 2021; 2023) of a given machine learning model—that is, a simple, high-level model that predicts how an intricate, low-level process (in this case, the computation graph G_M) behaves.

B. Discussion

In this section, we put component modeling in context with work in mechanistic interpretability and model editing. We also discuss some key limitations of COAR.

How does component attribution differ from mechanistic interpretability? In one sense, component attribution falls under the realm of mechanistic interpretability (e.g., Vig et al. (2020); Casper et al. (2022); Wang et al. (2022); Meng et al. (2022)) since our goal is indeed to understand how models internally process examples. However, our approach differs from a typical "mechanistic approach" in that rather than attempting to find circuits for a specific capability or uncovering the function of a specific component, component modeling takes a top-down, capability-agnostic perspective. That is, our main goal is to find a proxy for model behavior on a specific example as a function of model components. Analyzing this proxy then turns out to be a reliable way of editing models and uncovering subpopulations, as shown in Section 5. The top-down nature of our approach makes COAR immediately scalable to large models, and our focus on specific examples rather than human-prescribed capabilities eliminates some subjectivity (and inevitable misspecification) from the method itself, deferring it instead to a deliberate human decision. Furthermore, recent work (Wen et al., 2023; Friedman et al., 2023) demonstrates that bottom-up mechanistic analyses that solely analyze specific model components or its hidden representations in isolation can lead to misleading conclusions about model behavior.

Does localization help with model editing? The extent to which localizing specific model behavior to a subset of model components helps with model editing remains contested. On one hand, Hase et al. (2023) show that localizing factual associations in language models does not necessarily help with editing these associations. Additional evaluation studies show that model edits can fail to consistently modify model behavior as targeted (Cohen et al., 2023) and degrade robustness to distribution shifts (Brown et al., 2023). On the other hand, recent work shows that localization methods can in fact recover "ground truth" localization in controlled settings (Chang et al., 2023) and improve calibration of fine-tuned language models (Panigrahi et al., 2023). Our findings in Section 5 substantiate the latter view, as COAR-EDIT directly enable model editing in a variety of settings. Based on these findings, we hypothesize that the effectiveness of localization methods for model editing (a) depends on the causal efficacy of the localization method itself and (b) the intrinsic difficulty of different editing tasks.

Limitations. Our proposed method for estimating component attributions, COAR, is not without its limitations. First, the major computation bottleneck in COAR is that constructing a component dataset for a given example requires a moderately large number of forward passes through the model. In Appendix G.5, we show that the sample size required to estimate component attributions can be reduced by 2-5$\times$ without significantly impacting the quality of the resulting attributions. Improving the sample efficiency of component attribution through better sampling or approximation techniques would further mitigate this bottleneck. Second, specifying the "right" computational graph for a given task can be tricky. For example, a computation graph over neurons rather than over attention heads would lead to finer-grained localization, and thus better model editing, but would also make estimating component attributions more expensive. Similarly, COAR requires a choice of ablation method (Equation 1). While we use zero ablations due to its simplicity (Remark 2), more sophisticated ablations (e.g., Chan et al. (2022)) may be more appropriate for different tasks and/or model architectures (Zhang & Nanda, 2023). In Appendix G.4, we show that COAR is not dependent on the zero-ablation method and can be used with an alternative ablation method that simply scales down the activations of ablated components by a constant factor. Finally, while we extensively test the effectiveness of COAR in editing model behavior, we do so in a proof-of-concept manner. Developing finer-grained editing methods that leverage component attributions as a building block is an interesting avenue for future work.

C. Future work

Below, we highlight a few directions that, while outside the scope of this work, may be interesting avenues for future work to explore.

Attributing generative models. In this work, we focus our study to image classifiers. However, COAR is a general method in that given an appropriate model output function, it can estimate component attributions for any given machine learning model. Future work might thus explore possible model output functions (and their corresponding component attributions) for *generative* models. For diffusion-based generative models, one might study the denoising error for a fixed timestep, as in (Georgiev et al., 2023; Zheng et al., 2023). For language models, a possible point of start (following Park et al. (2023)) would be to use the average correct-class margin (5) of a sequence of tokens as the model output function. In fact, our preliminary experiments in Appendix F demonstrate that COAR yields predictive component attributions for GPT-2 (Radford et al., 2019) and Phi-2 (Javaheripi & Bubeck, 2023) without requiring any modifications to the algorithm. In general, estimating and applying component attributions for generative models is a promising avenue for future work.

Beyond linear attributions. Recall from Definition 2 that component attribution is a linear instantiation of the component modeling task (Definition 1). Linearity makes component attributions rather interpretable, and our results (Section 4) indicate that component attributions can still accurately predict model behavior. Still, the fact that component attributions' predictiveness decreases on out-of-distribution component subsets (i.e., when $\alpha_{\text{test}} \neq \alpha_{\text{train}}$) suggests that linear models might not be expressive enough to fully capture the map between model components and outputs. An potential avenue for future work would thus be to explore other (non-linear) model classes that map between ablated components and model output (e.g., decision trees or kernel methods). Note that the generality of COAR allows one to learn component models for any model class of choice.

Studying neural network representations. Finally, another interesting direction for future work would be to use component attribution (and component models, more generally) to study empirically documented phenomena in deep learning. There are a plethora of questions to ask here which, although beyond the scope of this work, are natural extensions of the results here. For example, extending our results from Section 5.1, can we use component attribution to better isolate "conflicting features" (Ilyas et al., 2022; Rosenfeld & Risteski, 2023) for a given task, and to understand their role in the training process (Rosenfeld & Risteski, 2023)? Can we study redundancy in how concepts are represented by neural networks, and how this representation evolves over the course of training? Similarly, can we develop improved methods for localizing memorized inputs to specific model components (Feldman & Zhang, 2020; Maini et al., 2023)? Given that component attributions are causally meaningful, can we use them as a kernel with which to compare different models (Kornblith et al., 2019) or learning algorithms (Shah et al., 2023)? Relatedly, are component models transferable across tasks (allowing us to view them as sparse "subpopulation vectors" (Ilharco et al., 2022))?

D. Additional COAR-EDIT experiments

D.1. "Forgetting" a class

We now consider "selective forgetting" problem (Wang et al., 2023), where the goal is to impair model performance on (only) a specific set of examples. In this experiment, we edit the same ImageNet ResNet-50 classifier (Setup B) as in Section 5.1, with the goal of forgetting the entire "chain-link fence" class. Like before, we use our editing approach COAR-EDIT (see (8) and (9)) to identify components that, when ablated, decrease the model's correct-class margin on examples from the "chain-link fence" class, but not on reference examples from other classes.

Results. Figure 6 summarizes our findings. In Figure 6a, we show that ablating just eight (out of $22,720$) model components degrades accuracy on the "chain fence" class from 66% to 20% while preserving overall accuracy on the train and validation set. Then, in Figure 6b, a comparison of class-wise accuracies before and after the edit shows that our approach specifically targets the "chain fence" class without impacting performance on any other class. Finally, Figure 6c uses the ImageNet-Sketch (Wang et al., 2019) (top) and ImageNet$\star$ (Vendrow et al., 2023) (bottom) datasets to show that the our edit is robust to distribution shifts in both the target and reference distribution.

Through additional experiments in Appendix H.2, we highlight that (a) our approach is sample-efficient, not needing many samples from the target and reference distributions to find effective edits; and (b) our findings are robust to the choice of class to forget.

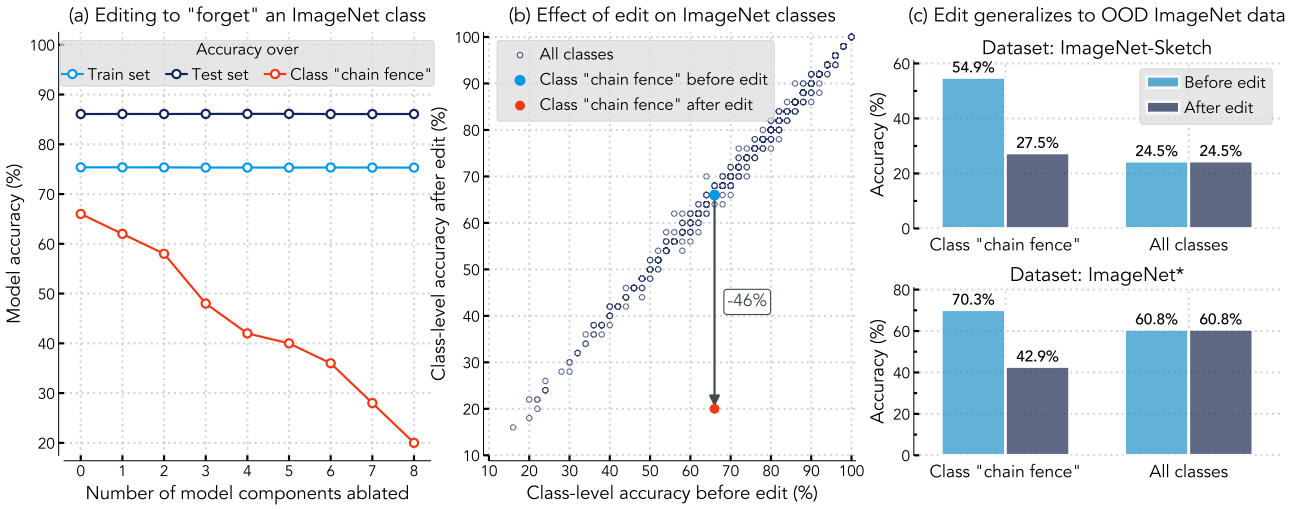

Figure 6: *"Forgetting" a class with* COAR-EDIT. We edit an ImageNet-trained ResNet-50 (Setup B from Section 4) to forget the "chain-link fence" class. On the left, we show how model accuracy on the class of interest (red) degrades as a function of k (the number of components removed), while model accuracy on the train (blue) and test (black) sets remains constant. In the center panel, we show the per-class accuracy of the model before and after an edit with $k = 8$ components—while accuracy on the chain-link fence class degrades significantly, accuracy on other classes stays roughly constant. Finally, on the right we evaluate the effects of the edit on class-specific and overall accuracy for distribution-shifted versions of ImageNet (namely ImageNet-Sketch (Wang et al., 2019) and ImageNet$\star$ (Vendrow et al., 2023)). As desired, our edit has a significant effect on the chain-link fence class, while leaving average model performance unchanged.

D.2. Mitigating a backdoor attack

We now use COAR to edit a model in order to reduce its sensitivity to backdoor attacks (Biggio et al., 2012; Gu et al., 2017). In a backdoor attack, an adversary introduces an artificial correlation into the training dataset of a machine learning model, causing the resulting model to rely on a spurious signal at test time. (For example, Gu et al. (2017) place a small square in the top corner of a random subset of training examples and relabel them with the "horse" class—models trained on this dataset will label *any* image containing a square as a horse.)

Experiment setup. In this experiment, our goal is to edit a model and remove its dependence on a spurious backdoor feature. We consider a ResNet18 with a computation graph that comprises all $2,344$ convolution filters that is trained on a modified CIFAR-10 dataset. Specifically, the dataset is "backdoored" in that an adversary has constructed a spurious correlation between a small blue-squared pattern and the "airplane" class (Figure 7a). As shown in Figure 7a, the resulting model latches on to the spurious pattern—simply adding the "airplane" trigger to CIFAR-10 test examples drops model accuracy from 89% (middle row) to 37% (bottom row).

Results. To edit this model, we apply COAR-EDIT over *paired* examples—i.e., examples with and without the backdoor trigger—to identify and ablate trigger-specific components. The trigger-specific components correspond to components that, when ablated, correct the misclassifications induced by the trigger without impacting predictions on test examples without the trigger. Figure 7b shows that ablating 25 components (1%) is sufficient to boost model accuracy on test examples with the trigger (red) from 37% to 84%—a 47% improvement. Furthermore, the model edit does not degrade accuracy on test examples without the trigger (blue) by more than 1%. In Figure 7c, we compare how model outputs on paired test examples with the trigger (y-axis) and without the trigger (x-axis) correlate before (top) and after the edit (bottom). Both subplots show that the model edit suppresses the effect of the trigger even at the example level, improving correlation between model outputs on examples with and without the trigger from 0.41 to 0.92. We also note that our approach is sample-efficient, requiring only 5 paired target and reference examples with and without the trigger to effectively "remove" the trigger via model editing. We defer additional details to Appendix H.4.

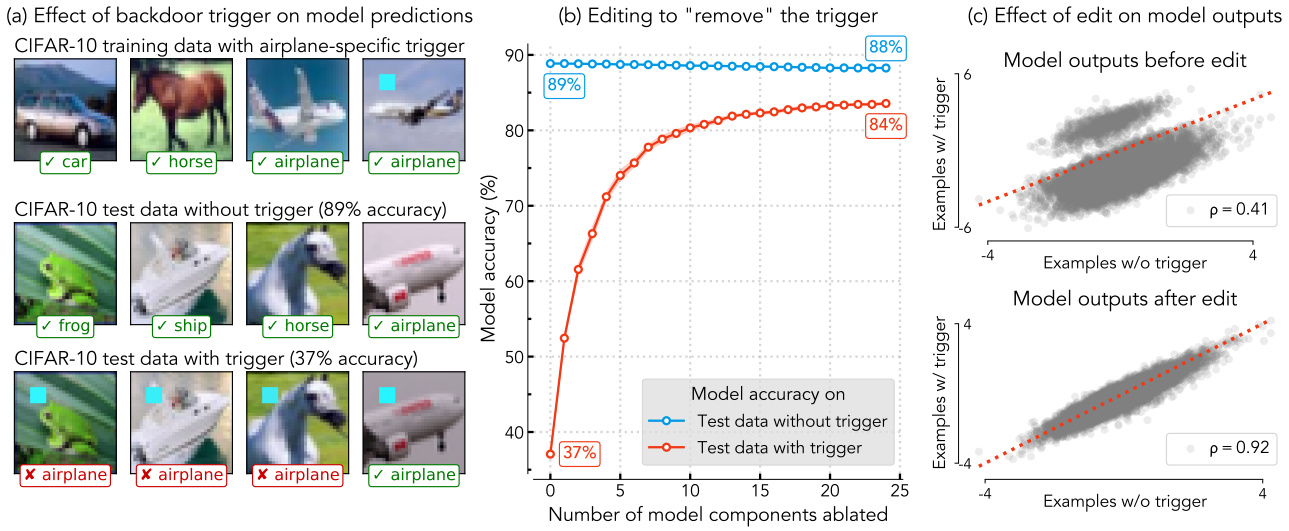

Figure 7: *Mitigating backdoor attacks with* COAR-EDIT. We edit a ResNet18 trained on a backdoored CIFAR-10 dataset to remove its dependence on a planted blue-squared trigger that is spuriously correlated with the "airplane" class. On the left, we show that the model is sensitive to the trigger—adding the blue trigger to CIFAR-10 test examples drops model accuracy from 89% (middle row) to 37% (bottom row). In the center panel, we show that ablating 25 components (1%) is sufficient to boost model accuracy on test examples with the trigger (red) from 37% to 84% without impacting accuracy on test examples without the trigger (blue). On the right, we show that the model edit suppresses the effect of the trigger even at the example level—the correlation between model outputs on paired test examples with and without the trigger improves from 0.41 to 0.92.

E. Evaluation setup

In this section, we outline the experiment setup—datasets, models, baselines, implementation details—used in Section 4 to evaluate whether COAR attributions can accurately estimate ground-truth component counterfactuals.

E.1. Pseudocode

Algorithm 1 An outline for estimating component attributions with COAR.

procedure COAR(example z, model M with output function f_M and components C, subsampling frac. α)
  Set $D^{(z)} \leftarrow []$ ▷ *initialize component dataset* (2)
  for $i \in \{1, \ldots, m\}$ **do** ▷ *m denotes dataset size*
    Sample a subset $C_i \subset C$ from $\mathcal{D}_C$ where $|C_i| = \alpha \cdot |C|$
    Set $y_i \leftarrow f_M(z, C_i)$ ▷ *compute component counterfactual* (1)
    Define $\mathbf{1}_{C_i} \in \{0,1\}^{|C|}$ as $(\mathbf{1}_{C_i})_j = 1$ if $c_j \in C_i$ else 0
    Update $D^{(z)} \leftarrow D^{(z)} + [(\mathbf{1}_{C_i}, y_i)]$ ▷ *update component dataset*
  $\theta^{(z)}, b^{(z)} \leftarrow$ LINEARREGRESSION($D^{(z)}$) ▷ *estimate component attributions via Equation* 3
  return $\theta^{(z)}, b^{(z)}$

Figure 8: Pseudocode for estimating component attributions with COAR.

E.2. Datasets and models.

We now outline the datasets and models used to evaluate COAR (§4) and COAR-EDIT (§5).

CIFAR-10. We use the standard CIFAR-10 (Krizhevsky, 2009) image classification dataset to evaluate COAR attributions (Section 4, Appendix G.2) and for an editing task (Appendix D.2). We train ResNet, ViT, and MLP models that attain test accuracies of 91%, 83% and 56% respectively. We specify a computation graph over $2,344$ components for the ResNet-18 model, $31,728$ components for the ViT model, and $3,072$ components for the MLP model. Each component in the ResNet-18 model corresponds to a convolution filter. Similarly, each component in the ViT and MLP models corresponds to a neuron.

*ImageNet. We use the standard ImageNet (Deng et al., 2009) image classification dataset to evaluate COAR attributions in Section 4 and for editing tasks in Appendix D.1. We use ImageNet-Sketch (Wang et al., 2019) and five random shifts from ImageNet⋆ (Vendrow et al., 2023)—"in the water", "at dusk simple", "orange", "pencil sketch", "green"— to evaluate the out-of-distribution performance of edited ImageNet models in Appendix D.1. We use the pre-trained ResNet50 and ViT-B/16 models[2] that attain test accuracies of 75.4% and 80.7% respectively. For the ResNet-50 model, we specify a computation graph over $22,720$ components, each corresponding to a convolution filter. Similarly, for the ViT-B/16 model, we specify a computation graph over $82,944$ components, each corresponding to a neuron.

Waterbirds. The Waterbirds dataset (Sagawa et al., 2020) comprises images of birds taken from the CUB dataset (Wah et al., 2011) and pasted on backgrounds from the Places dataset (Zhou et al., 2017). The task here is to classify "waterbirds" and "landbirds" in the presence of spurious correlated "land" and "water" backgrounds in the training dataset. Sagawa et al. (2020) introduce Waterbirds as a benchmark to improve model performance under subpopulation shifts induced by spurious correlations. We use this dataset to evaluate whether COAR-EDIT can improve subpopulation robustness via model editing. In this experiment, we fine-tune an ImageNet ResNet50 model and use a computation graph over $22,720$ components, each corresponding to a convolution filter.

CelebA. The CelebA dataset (Li et al., 2020) comprises images of celebrities with binary attributes such as "smiling", "wearing hat", "wearing lipstick", etc. Similar to previous work on subpopulation robustness (e.g., (Sagawa et al., 2020)), we repurpose CelebA as a binary classification task where the goal is to predict whether a person in a given image has blond hair. The attributes "hair color" and "gender" are spuriously correlated in the training dataset, resulting in models that latch on to a "gender $\rightarrow$ blond hair" shortcut and underperform on the "blond males" subpopulation. Similar to the Waterbirds setting, we fine-tune an ImageNet ResNet50 model and specify a computation graph over $22,720$ components,

[2]Model and pre-trained weights taken from torchvision: https://pytorch.org/vision/stable/models.html

each corresponding to a convolution filter.

Typographic attacks dataset. We use a dataset of typographic attacks (Materzyńska et al., 2022) for an editing task in Section 5.3. This dataset comprises 180 images of household objects with and without eight typographic attacks such as "taxi", "twitter", "EU", and "iPad". We visualize some examples from this dataset in Figure 5. Our experiment in Section 5.3 uses this dataset along with a zero-shot CLIP ViT-B/16 classifier (Radford et al., 2021). For this model, we specify a computation graph over all $82,944$ components, corresponding to the set of all weight vectors (individual rows in weight matrices) in all self-attention and MLP modules. See Appendix H.5 for more details.

*TinyStories. We use the TinyStories dataset (Eldan & Li, 2023) to evaluate COAR attributions over the GPT-2 language model (Appendix F). This dataset contains short stories synthetically generated by GPT-3.5 and GPT-4. To compute component attributions for GPT-2, we specify a computation graph over $64,512$ components, which correspond to the set of all weight vectors, i.e., in every self-attention module and feed-forward module of the model. See Appendix F.1 for experiment details and findings.

*BoolQ. We use the BoolQ dataset (Clark et al., 2019) to evaluate COAR attributions for the Phi-2 model (Li et al., 2023b). Each example in this dataset comprises a passage of text, a question, and a binary answer. We evaluate the zero-shot performance of Phi-2 using the prompting and evaluation procedure from Gao et al. (2023)[3]. Given the size of the Phi-2 model, we specify a computation graph over $55,552$ components, each corresponding to a contiguous block of 10 weight vectors in every self-attention module and feed-forward module of the model. See Appendix F.2 for experiment details and findings.

E.3. Baselines

In Section 4, we compare COAR against four baseline methods for estimating component attributions: Leave-One-Out (LOO), Gradient-times-parameters (GP), Neuron Conductance (NC), and Internal Influence (II). Each baseline computes an attribution vector $\boldsymbol{w^{(z)}} \in \mathbb{R}^{|C|}$ for a given example z by assigning an "importance" score $w_j^{(z)}$ to each component $c_j \in C$. Then, as per Equation 4, we estimate a component counterfactual $f_M(z, C')$ as the sum of importance scores of components in $C \setminus C'$, i.e., scores of components that are not ablated. We describe each baseline in more detail below:

- **Leave-One-Out** (LOO): This method ablates each component $c_j \in C$ and sets the coefficient ${\theta^{(z)}}_j$ to the change in model output $f_M(z)$ before and after ablation:

$$ {w^{(z)}}_j = f_M(z, \{c_j\}) - f_M(z, \emptyset) $$

- **Gradient-times-Parameters** (GP): This method approximates the leave-one-out estimate described above. Specifically, it estimates the leave-one-out effect of each component $c_j \in C$ using a first-order Taylor approximation of $f_M(z, \{c_j\})$ around $f_M(z, \emptyset)$:

$$ w_j^{(z)} = \nabla_{c_j} f_M(z, \emptyset) \cdot \delta_{c_j} $$

  where δ_{c_j} is the parameter-space change in c_j induced by the ablation method of choice.

- **Neuron Conductance** (NC) (Dhamdhere et al., 2018): This method extends the Integrated Gradients method (Sundararajan et al., 2017)—an input-space feature attribution method—to compute importance scores for each component $c_j \in C$. Intuitively, NC modifies the computation in Integrated Gradients in order to quantify the "flow" through each component $c_j \in C$. See Equation 3 in (Dhamdhere et al., 2018) for a formal description.

- **Internal Influence** (II) (Leino et al., 2018): Similar to NC, this method also adapts Integrated Gradients (Sundararajan et al., 2017) to compute importance scores. At a high level, II directly applies Integrated Gradients to layerwise activations by treating the output of each layer as an input to subsequent layers. See Definition 1 in (Leino et al., 2018) for a formal description.

We implement the first two baselines (LOO and GP) from scratch[4] and use the `captum` library (Kokhlikyan et al., 2020) [5] to implement NC and II. As per Definition 2, we estimate the component counterfactual $f_M(z, C')$ using these baselines by setting the bias term $b^{(z)}$ to zero and taking the sum over attribution scores of components that are not ablated.

[3] https://github.com/EleutherAI/lm-evaluation-harness/
[4] Our code is available at https://github.com/MadryLab/modelcomponents
[5] Github repository: https://github.com/pytorch/captum

E.4. Implementation details

Sample size for component attribution estimation. The computational cost of our approach linearly scales with the sample size m used to estimate component attributions (see Figure 8). Each sample in the component dataset $D^{(z)}$ corresponds to a single forward pass through the model M in order to compute the counterfactual $f_M(z, C')$ (1), i.e., model output $f_M(z)$ after ablating a subset of components $C' \subset C$. The setups $\{\text{A}, \text{B}, \text{C}\}$ considered in Section 4 use sample size $m = \{50000, 100000, 200000\}$ respectively. In Appendix G.5, we show that the sample size m used in Section 4 can be reduced by 2-5$\times$, resulting in a direct speedup while only reducing the predictive power of COAR attributions by a small amount.

Data loading. We use the `FFCV` library[6] (Leclerc et al., 2022) to train and evaluate models. `FFCV` removes the data loading bottleneck for small models, gives a 3-4$\times$ improvement in throughput compared to standard PyTorch data loading.

*Speeding up regression. The second step of COAR—fitting component attributions to the component dataset (2)—requires solving a linear regression problem (Equation 3) for each example z. We parallelize this step by using the `fast-l1` package[7], a SAGA-based GPU solver for linear regression.

Computing resources. We train our models and compute COAR attributions on a cluster of machines, each with 9 NVIDIA A100 or V100 GPUs and 96 CPU cores. We also use half-precision to increase training speed.

[6]Github repository: https://github.com/libffcv/ffcv
[7]Github repository: https://github.com/MadryLab/fast_l1

F. Applying COAR to language models

In Section 4 and Appendix G, we showed that our proposed method COAR attributions accurately estimate component counterfactuals (1) on large-scale vision tasks across several datasets and model architectures. In this section, we apply COAR to language models. Specifically, we consider two experiments: (a) GPT-2 (Radford et al., 2019) evaluated on the next-token prediction task and (b) Phi-2 (Li et al., 2023b) evaluated on a zero-shot classification task. In both cases, we show that COAR attributions accurately predict how model outputs change in response to component ablations.

F.1. Evaluating GPT-2 on the TinyStories dataset

Task and model output function. We apply COAR to the next-token prediction task. Following Park et al. (2023), we interpret this task as a sequence as a v-way classification problem, where v is the vocabulary size, and set the model output function to be the average correct-class margin (5) over all tokens in a given sequence.

Model and dataset. In this experiment, we consider the GPT-2 model[8] (Radford et al., 2019), with a computation graph over $64,512$ components. These components correspond to the set of weight vectors in every self-attention module and feed-forward module in the model. We evaluate model performance on the next-token prediction task using the TinyStories dataset[9] (Eldan & Li, 2023), where each sequence corresponds to a synthetically generated short story.

Computing COAR attributions. We apply COAR (without any modifications) to compute component attributions for a random subset of 1000 examples in the TinyStories validation set using a component dataset of $200,000$ component counterfactuals (2) and a ablation fraction of $\alpha = 2.5\%$.

Evaluating COAR attributions. Similar to the results in Section 4, COAR attributions are predictive in the language modeling setting as well. Specifically, these attributions accurately predict the effect of ablating components on the average correct-class margin of GPT-2 on examples from the TinyStories validation set. In Figure 9a, we pick a random example z from the TinyStories validation set and compute the correlation between ground-truth component counterfactuals $f_M(z, \cdot)$ and the corresponding estimate (4) using its COAR attributions $\boldsymbol{\theta}^{(z)}$, as defined in Equation 6. In Figure 9b, we plot a histogram over example-level correlations of 1000 examples and find that COAR attributions attain an average correlation of $\{0.83, 0.85, 0.89\}$ with ground-truth component counterfactuals sampled using ablation fraction $\alpha = \{5\%, 2.5\%, 1\%\}$ respectively.

F.2. Evaluating Phi-2 on the BoolQ dataset

Task and model output function. We now turn to a reading comprehension task, where the goal is to answer a question given a passage of text. We evaluate this classification task in a zero-shot manner: the language model is prompted with a passage of text and a question, and the goal is to output the correct answer from {yes, no}. Like in vision tasks (Section 4), we use the correct-class margin (5) as the model output function for this zero-shot binary classification task.

Model and dataset. We consider the Phi-2 model[10] (Li et al., 2023b) and specify a computation graph over $55,552$ components. Here, each component corresponds to a contiguous block of 10 weight vectors in the model. We evaluate this model on the BoolQ dataset[11] (Clark et al., 2019), where each example consists of a passage of text, a question, and a binary {yes, no} answer. Using the prompting and evaluation procedure from the Gao et al. (2023)[12], Phi-2 attains an 83.6% accuracy on this task.

Computing COAR attributions. Like in Appendix F.1, we apply compute COAR attributions for a random subset of 500 examples in the BoolQ validation set using a component dataset of $m = 100,000$ component counterfactuals (2) and a ablation fraction of $\alpha = 0.025$.

Evaluating COAR. We find that COAR attributions are predictive of unseen component counterfactuals on this task as well. Figure 10a plots the correlation between ground-truth component counterfactuals $f_M(z, \cdot)$ and the corresponding COAR estimate (4) of a random BoolQ example z. The histograms in Figure 10b show that COAR attributions attain correlation $\{0.58, 0.66, 0.66\}$ with component counterfactuals sampled using ablation fraction $\alpha = \{5\%, 2.5\%, 1\%\}$ respectively.

[8] https://huggingface.co/gpt2
[9] https://huggingface.co/datasets/roneneldan/TinyStories
[10] https://huggingface.co/microsoft/phi-2
[11] https://huggingface.co/datasets/google/boolq
[12] https://github.com/EleutherAI/lm-evaluation-harness

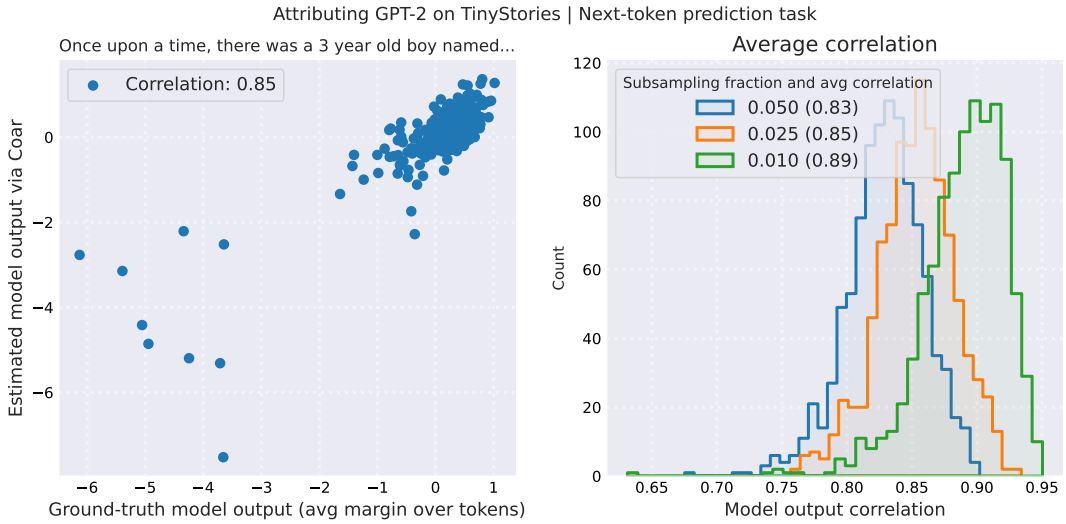

Figure 9: Evaluating COAR on GPT-2. We apply COAR to the GPT-2 model (Radford et al., 2019) on the TinyStories dataset (Eldan & Li, 2023). The resulting component attributions are predictive of component counterfactuals. The left plot shows that component attributions can estimate the effect of ablating components on the average correct-class margin (over tokens in a sequence) of GPT-2 on a random TinyStories example with high correlation. The histograms in the right plot show that COAR attributions attain high average correlation for multiple values of ablation fraction α.

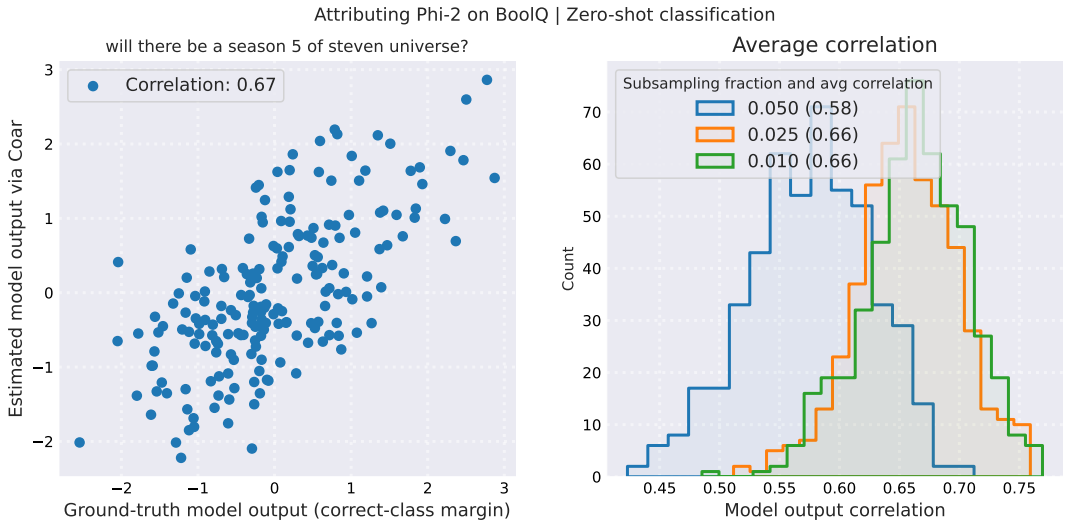

Figure 10: Evaluating COAR on Phi-2. We apply COAR to the Phi-2 model (Javaheripi & Bubeck, 2023) on the BoolQ dataset (Clark et al., 2019). The resulting component attributions are predictive of component counterfactuals. The left plot shows that component attributions can estimate the effect of ablating components on the average correct-class margin of Phi-2 on a random BoolQ example with high correlation. The histograms in the right plot show that COAR attributions attain high average correlation for multiple values of ablation fraction α.

G. Additional evaluation of COAR

In this section, we first show that COAR learns accurate component attributions on additional datasets, model architectures, and tasks (Appendices G.1 to G.3). This supplements our findings in Section 4, where we showed that COAR learns component attributions that accurately predict component counterfactuals (1) on three image classification setups: CIFAR-10 ResNet-18, ImageNet ResNet-50, and ImageNet ViT-B/16. Then, we show that COAR attributions retain its predictive power when estimated with fewer samples (Appendix G.5) or with different ablation fractions (Appendix G.4). Finally, we supplement our example-level evaluation of COAR attributions in Section 4 with additional example-level comparisons of ground-truth component counterfactuals and attribution-based estimates (Appendix G.6).

G.1. Evaluating COAR on additional datasets

Our experiments in Section 4 evaluated the predictiveness of COAR attributions corresponding to in-distribution test examples from the CIFAR-10 and ImageNet datasets. Now, we show that COAR attributions remain predictive on training examples as well as out-of-distribution examples. Specifically, we apply COAR to compute attributions of ResNet-18 predictions on the CIFAR-10 training set and on six corrupted versions of the CIFAR-10 test set (Hendrycks & Dietterich, 2019). as shown in Figure 11, COAR attributions exhibit high correlation on average (between 0.6 and 0.8) depending on the ablation fraction α used to ablate random α-fraction sized components subsets. Note that the correlation is maximum when $\alpha = 0.05$ because the component attributins are estimated with the same ablation fraction, i.e., $\alpha_{\text{train}} = 0.05$.

G.2. Evaluating COAR on additional model architectures

Recall that COAR is model-agnostic in that it is not tied to any specific model architecture. In Section 4, we applied COAR to ResNets trained on CIFAR-10 and ImageNet and a ViT-B/16 model trained on ImageNet. In this section, we apply COAR to two additional model architectures: a ViT model trained on CIFAR-10 (83% accuracy) and a one-layer fully-connected network trained on CIFAR-10 (56% accuracy). Figure 12 shows that COAR attributions on both architectures yield accurate estimates of how model outputs change in response to ablating random α-fraction sized components subsets, with correlation 0.65 and 0.85 for the ViT and MLP models when $\alpha = \alpha_{\text{train}}$ respectively.

G.3. Evaluating COAR on additional tasks

We now evaluate COAR attributions on four additional tasks:

- First, we apply COAR to pre-trained ImageNet ResNet50 model fine-tuned on two datasets—Waterbirds and CelebA—that we use in Section 5.2—see first row of Figure 13. We find that COAR attributions are predictive on both datasets, attaining higher correlation with ground-truth component counterfactuals when α is closer to $\alpha_{\text{train}} = 0.05$.
- Second, we apply COAR to a pre-trained ImageNet ResNet50 model fine-tuned on MIMIC-CXR (Johnson et al., 2019), a dataset of labeled chest radiographs. In this case, we set the model output function to be the logit of the "Cardiomegaly" class instead of correct-class margin that we use in Section 4. Figure 13 shows that COAR attributions attain a correlation of 0.7 and 0.6 with ground-truth logits when $\alpha = \alpha_{\text{train}} = 0.05$ and $\alpha = 0.10$ respectively.
- The fourth plot in Figure 13 corresponds to the CLIP setting considered in Section 5. In this setting, we take the zero-shot CLIP ViT-B/16 classifier and evaluate it on a dataset of images with and without typographic attacks (Materzyńska et al., 2022). As shown in the plot, the correlation between COAR attributions and ground-truth margins is close to 0.7 when $\alpha = \alpha_{\text{train}} = 0.03$, i.e., ablating 3% of the components in the CLIP model.

G.4. Comparing COAR attributions estimated with different ablation fractions

We now analyze how changing the ablation fraction α_{train} used to fit COAR attributions affects their predictiveness over different ablation fractions at test time. Specifically, we consider the ImageNet ResNet-50 setting from Section 4 and compute two sets of COAR attributions, corresponding to two values of α_{train}: 0.05 and 0.10. Then, for each of these two sets of attributions, we evaluate its correlation with ground-truth component counterfactuals over a range of ablation fractions α. As shown in Figure 14, the correlation "profile" over α depends on the value of α_{train} used to fit the attributions. When α is small, the correlation is higher for attributions estimated with $\alpha_{\text{train}} = 0.05$. Analogously, when α is large, the correlation is higher for attributions estimated with $\alpha_{\text{train}} = 0.10$. This is because the component attributions fare better

as counterfactual predictors on component counterfactuals that are "similar" to the ones used to fit them—i.e., when $\alpha_{\text{test}} \approx \alpha_{\text{train}}$.

G.5. Comparing COAR attributions estimated with different sample sizes

In Section 4, we computed COAR attributions using sample sizes $m = 50000$ for the ResNet-18 model trained on CIFAR-10 and $m = 100000$ for the ResNet-50 model trained on ImageNet. Recall that the sample size m here corresponds to the number of component counterfactuals used to fit the component attributions. In this section, we vary the sample size m and show that COAR attributions remain predictive even when trained on $k\times$ fewer examples, where $k \in \{2, 5, 10\}$. Specifically, the left column of Figure 15 shows that COAR attributions estimated on CIFAR-10 and ImageNet data with sample size m and m/k have high cosine similarity on average, with the similarity increasing as k decreases. The right column of Figure 15 shows that decreasing the sample size m by a factor of $k \in \{2, 5, 10\}$ does not significantly impact the correlation between COAR attributions and ground-truth component counterfactuals. For example, reducing the sample size by $5\times$ only reduces the correlation from 0.7 to 0.65 in the CIFAR-10 ResNet-18 setting. Additionally, we observe that COAR attributions fare better than attributions estimated with the best-performing baseline (`LOO`) even when trained on $10\times$ fewer examples on CIFAR-10 and $5\times$ fewer examples on ImageNet.

G.6. Analyzing COAR attributions at the example level

To supplement our evaluation in Section 4, we provide additional example-level scatterplot comparisons between ground-truth component counterfactuals and the corresponding estimates obtained using component attributions estimated with COAR and all baselines from Section 4. We plot these comparisons on CIFAR-10 examples in Figure 16 and on ImageNet examples in Figure 17. Our findings further substantiate that COAR attributions exhibit higher correlation with ground-truth component counterfactuals than all four baseliens on both CIFAR-10 and ImageNet.

G.7. Qualitatively analyzing COAR attributions

We qualitatively analyze COAR attributions using two visualization techniques:

Visualizing component-specific attributions across examples. Given examples $\{z_1, \ldots, z_n\}$ with corresponding component attributions $\{\theta^{(z_1)}, \ldots, \theta^{(z_n)}\}$, we analyze how the attribution estimates of individual components vary across the set of examples. Specifically, for a component $c_i \in C$, we visualize the examples with the most positive attribution values $\theta_i^{(z)}$ for component c_i. In this experiment, we visualize a random subset of components from the ImageNet ResNet-50 model (setup B in Section 4). As shown in Figure 18, the examples with the most positive attributions for a given component exhibit high visual similarity at different levels of granularity:

- The first, third and fifth row in Figure 18 show that the examples with the most positive attributions for `layer4.0.conv3[477]` and `layer4.2.conv3[53]` contain purple flowers, watch faces, and glass-shaped objects respectively.
- However, consistent with recent work on superposition in deep networks (Elhage et al., 2022), we observe that some components such as `layer4.2.conv2[336]` in the second row as well as `layer3.1.conv3[655]` in the last row can surface dissimilar subsets of examples and do not readily map to a single semantic concept.

Visualizing nearest neighbors in attribution space. We also use component attributions as feature embeddings in order to visualize the nearest neighbors of a given example in "component attribution" space. Intuitively, this technique allows us to identify examples on which model outputs change similarly in response to component ablations. In this experiment, we visualize a random subset of examples from the CelebA dataset along with their 5 nearest neighbors using COAR attributions of a fine-tuned ImageNet ResNet-50 model. Figure 19 shows that the nearest neighbors of a given example in attribution space high visual similarity, i.e., similar facial attributes such as background (first row), hair color (second and fourth row), accessories (third row), or even the same person in different poses (last row).

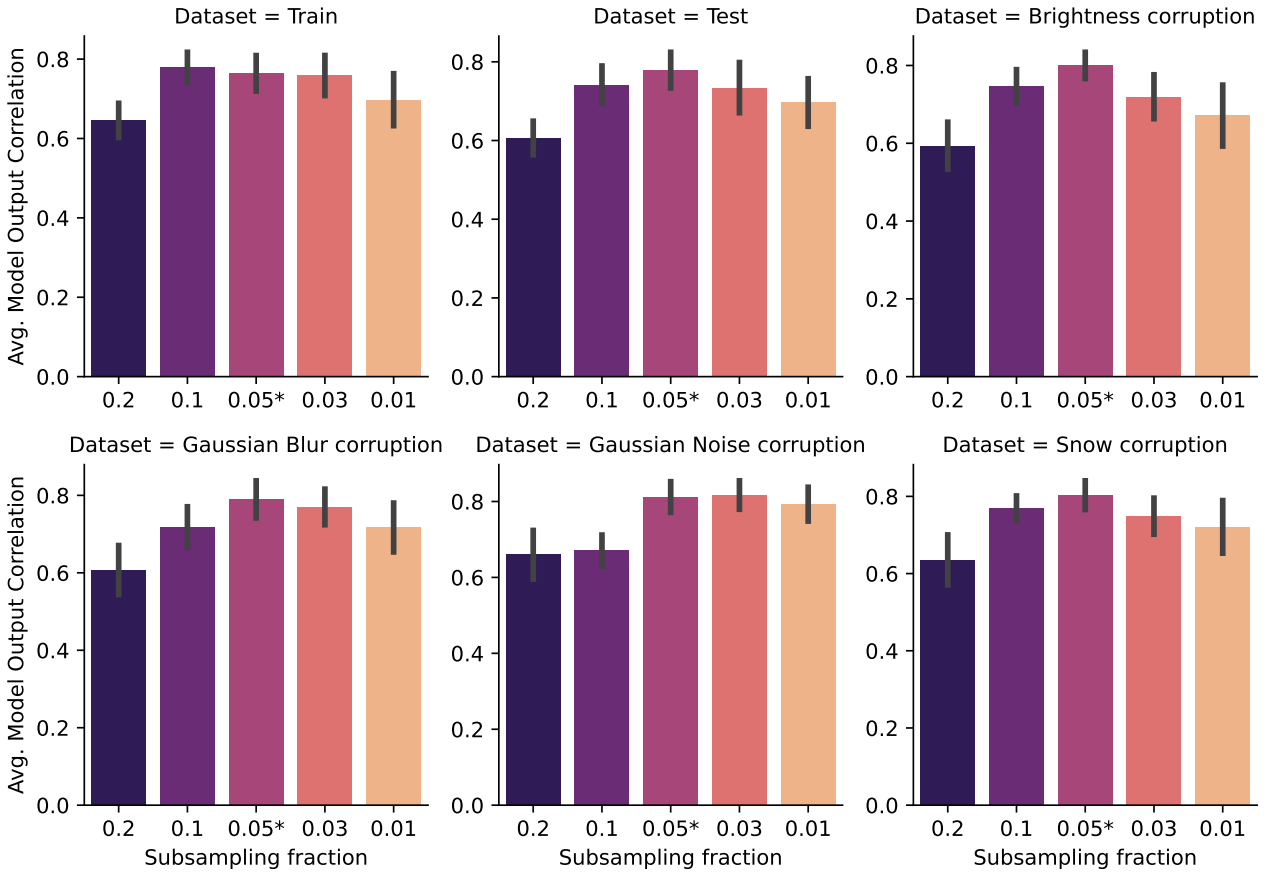

Figure 11: Do COAR attributions generalize to out-of-distribution examples? COAR attributions remain predictive on the CIFAR-10 training set and on six corrupted versions of the CIFAR-10 test set (Hendrycks & Dietterich, 2019) over a range of ablation fractions α. See Appendix G.1 for more details.

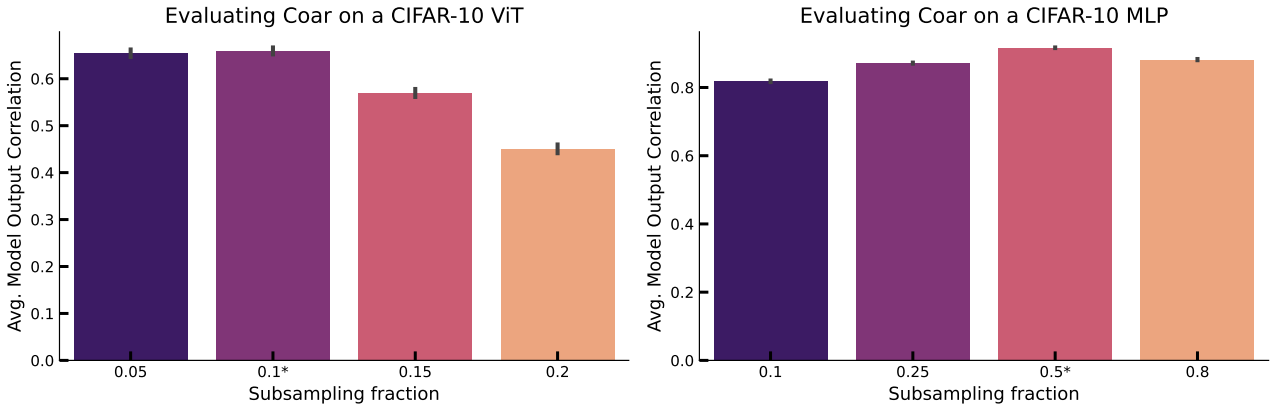

Figure 12: Do COAR attributions generalize to other model architectures? COAR attributions yield accurate estimates of component counterfactuals on two additional model architectures: a ViT-based model (left) and a one-layer fully-connected network (right) trained on CIFAR-10. See Appendix G.2 for more details.

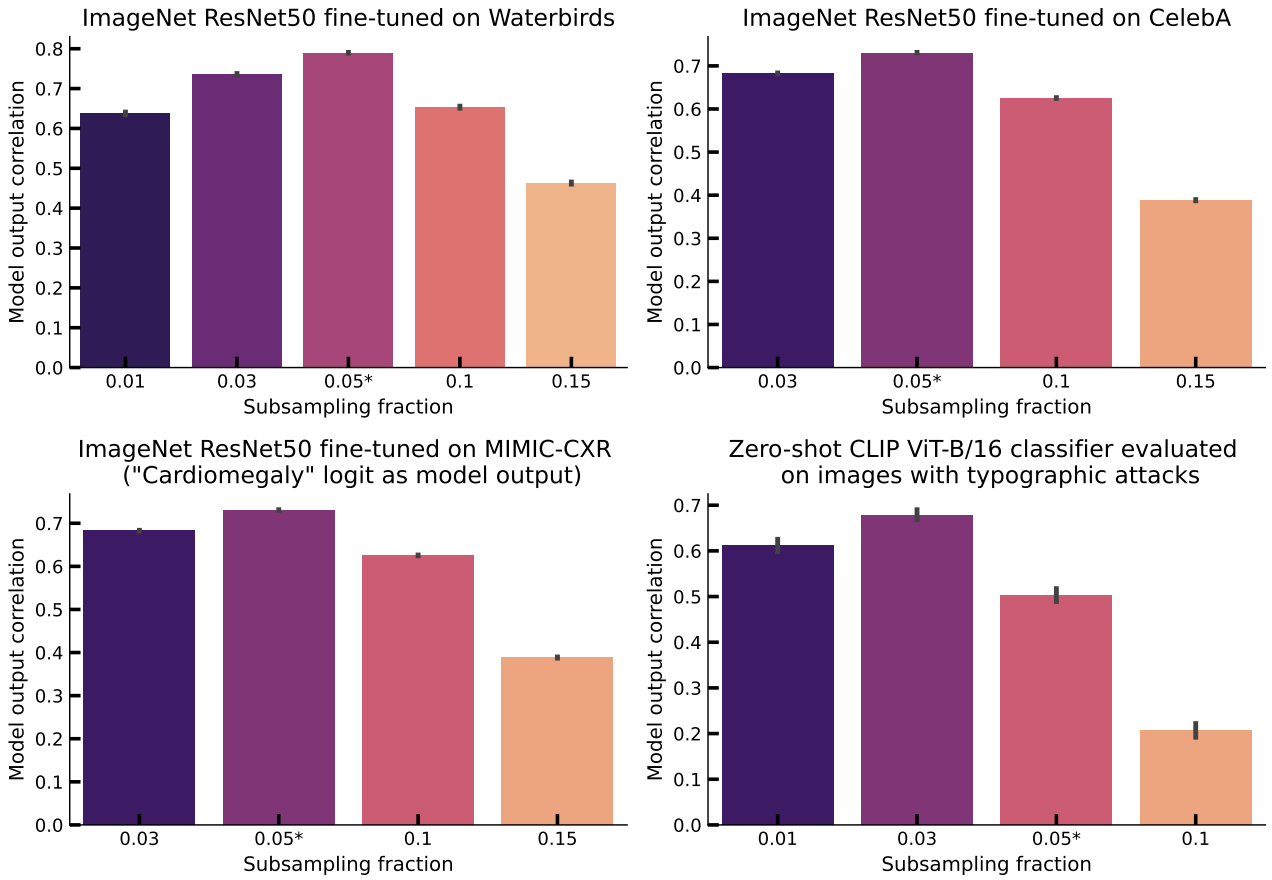

Figure 13: Evaluating COAR attributions on additional tasks. We find that component attributions estimated using COAR are predictive on four additional tasks: fine-tuning ImageNet ResNet50 on Waterbirds, CelebA and MIMIC-CXR, and a zero-shot CLIP ViT-B/16 classification task on a dataset containing typographic attacks (Section 5.3). Note that the MIMIC-CXR setting uses the logit of the "Cardiomegaly" class as the model output function. See Appendix G.3 for additional information about these tasks.

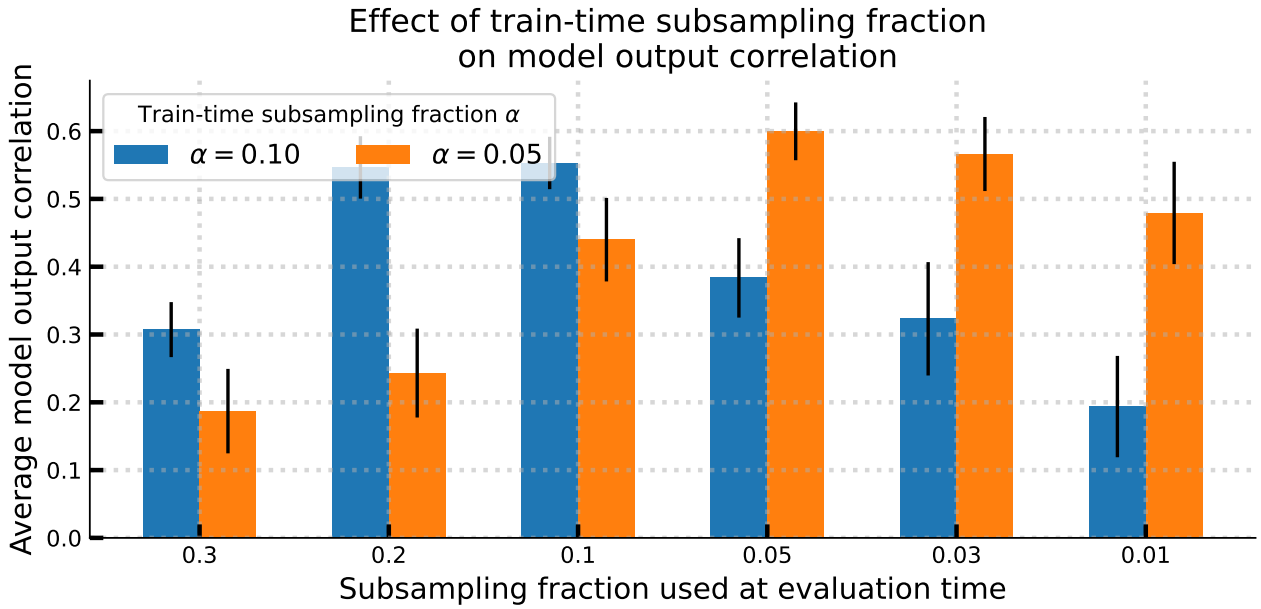

Figure 14: Comparing COAR attributions estimated with different ablation fractions α. COAR attributions estimated with different ablation fractions α_{train} attain a different correlation "profile" over α at test time. The correlation between ground-truth component counterfactuals and attribution-based estimates is higher for attributions estimated with $\alpha_{\text{train}} = 0.05$ when α is small, and higher for attributions estimated with $\alpha_{\text{train}} = 0.10$ when α is large. This empirically shows that COAR attributions are more predictive on component counterfactuals that are "similar" to the ones used to fit them—i.e., when $\alpha_{\text{test}} \approx \alpha_{\text{train}}$. See Appendix G.4 for more details.

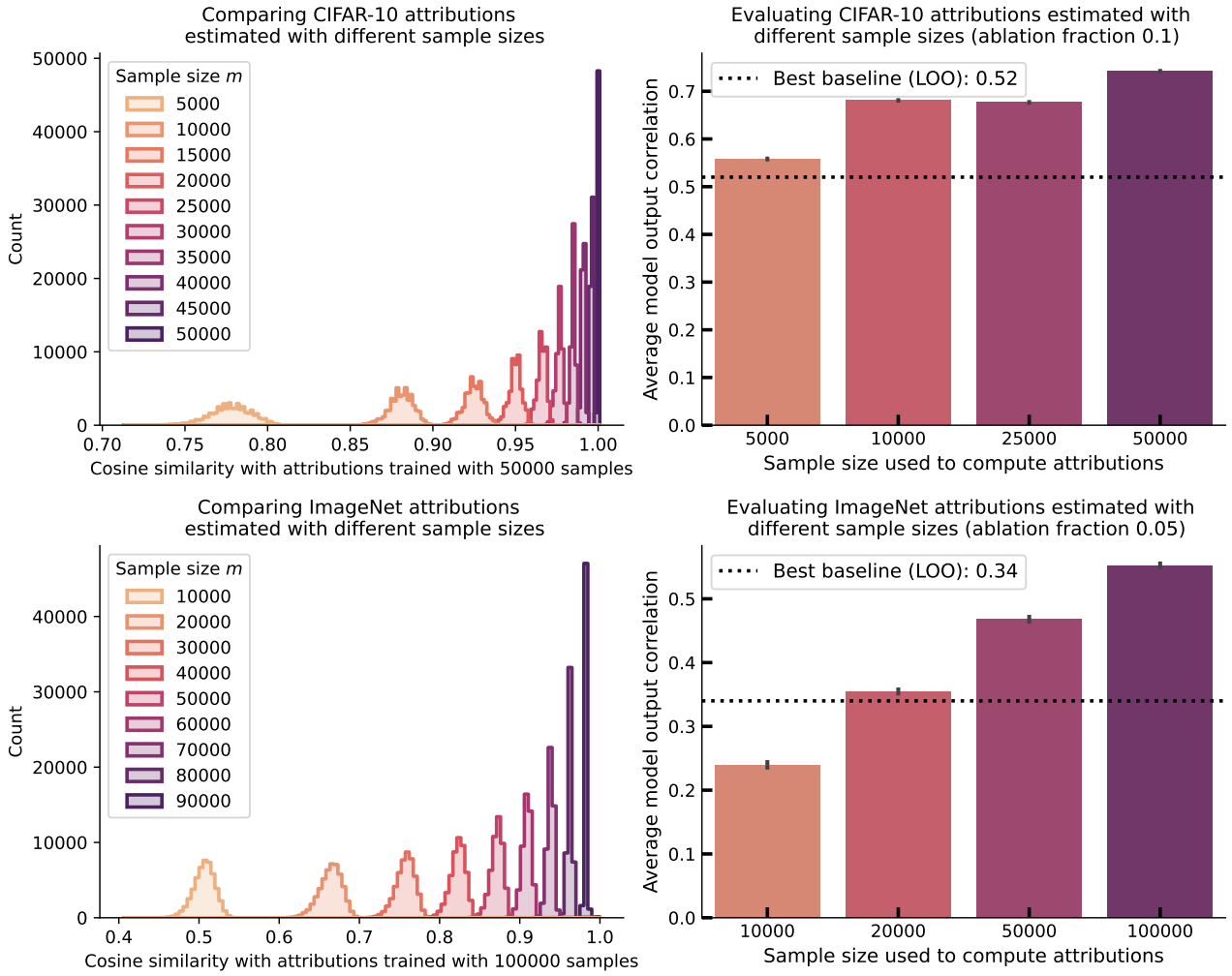

Figure 15: Comparing COAR attributions estimated with different sample sizes. COAR attributions for CIFAR-10 ResNet-18 and ImageNet ResNet-50 (Setup A and B respectively in Section 4) estimated with smaller sample sizes m are still predictive of component counterfactuals. On the left, we show that COAR attributions estimated with sample size m and m/k have high cosine similarity on average, with the similarity increasing as k decreases. On the right, we show that decreasing the sample size m by a factor of $k \in \{2, 5, 10\}$ does not significantly affect the correlation between COAR attributions and ground-truth component counterfactuals. In particular, COAR outperforms the best-performing baseline (LOO) even with $10\times$ fewer samples on CIFAR-10 (top row) and $5\times$ fewer samples on ImageNet (bottom row).

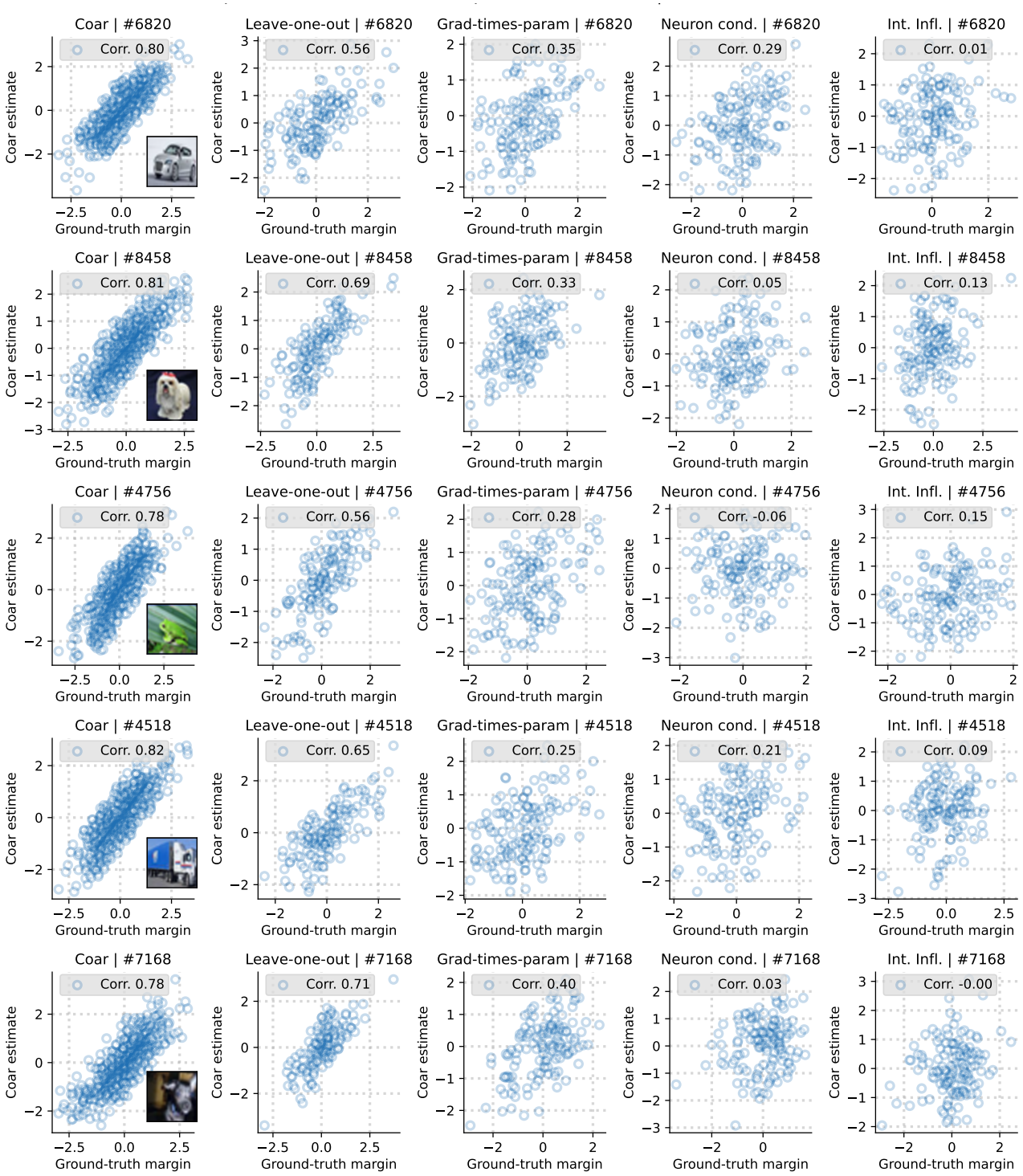

Figure 16: Additional example-level evaluation of component attributions on CIFAR-10. Each row corresponds to a different example z randomly picked from the CIFAR-10 test set and each column corresponds to a different attribution method. The left-most subfigure in each row shows that COAR attributions and the corresponding ground-truth component counterfactuals exhibit high correlation on example z. In comparison, the other subfigures in each row, one for baseline method, consistently exhibit lower correlation. See Appendix G.5 for more details.

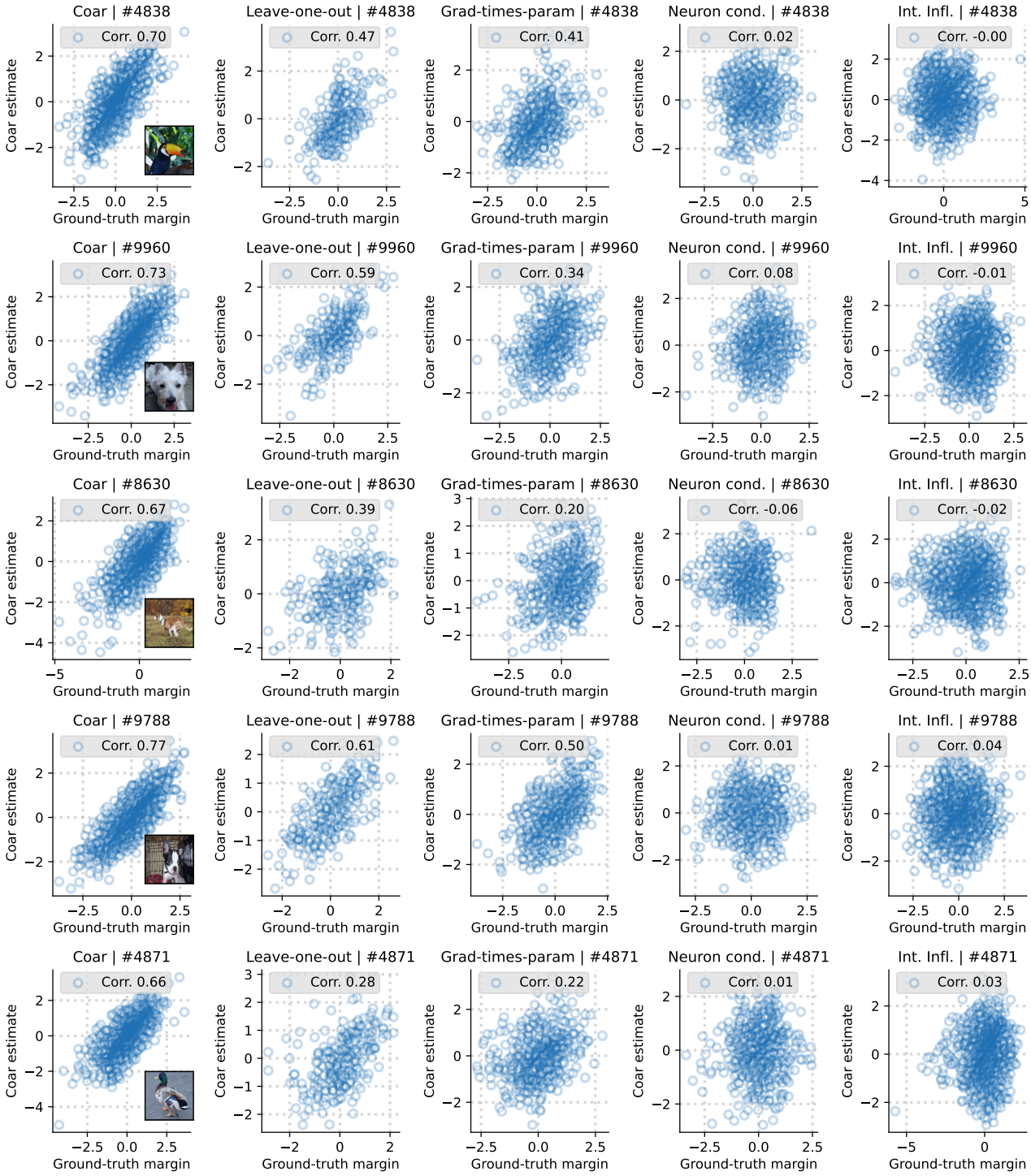

Figure 17: Additional example-level evaluation of component attributions on ImageNet. Similar to the results in Figure 16, each row corresponds to a different example z randomly picked from the ImageNet test set. The left-most subfigure in each row shows that COAR attributions and the corresponding ground-truth component counterfactuals exhibit high correlation on example z. In comparison, the other subfigures in each row, corresponding to a baseline method, consistently exhibit worse correlation. See Appendix G.5 for more details.

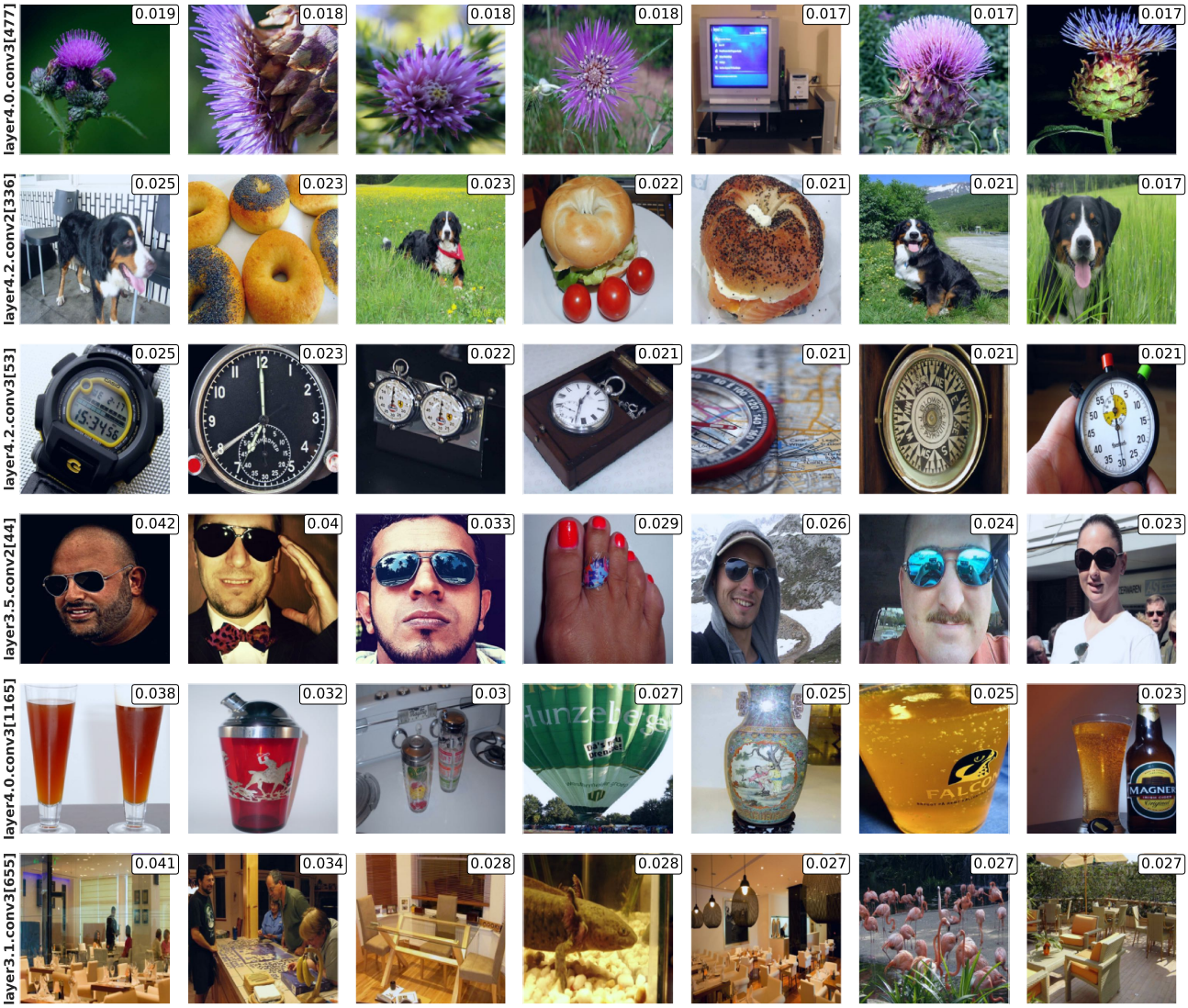

Figure 18: Visualizing component-specific attributions across examples. We sample a random set of components from the ImageNet ResNet-50 model (setup B in Section 4) and visualize the examples with the most positive attributions for each component. In general, the examples with the most positive attributions for a given component exhibit visual similarity at different levels of granularity. For example, the first, third and fifth row in Figure 18 show that the examples with the most positive attributions for `layer4.0.conv3[477]` and `layer4.2.conv3[53]` contain purple flowers, watch faces, and glass-shaped objects respectively. However, consistent with recent work on superposition in deep networks (Elhage et al., 2022), we observe that some components such as `layer4.2.conv2[336]` (second row) and `layer3.1.conv3[655]` (last row) can surface dissimilar subsets of examples or do not readily map to a single semantic concept.

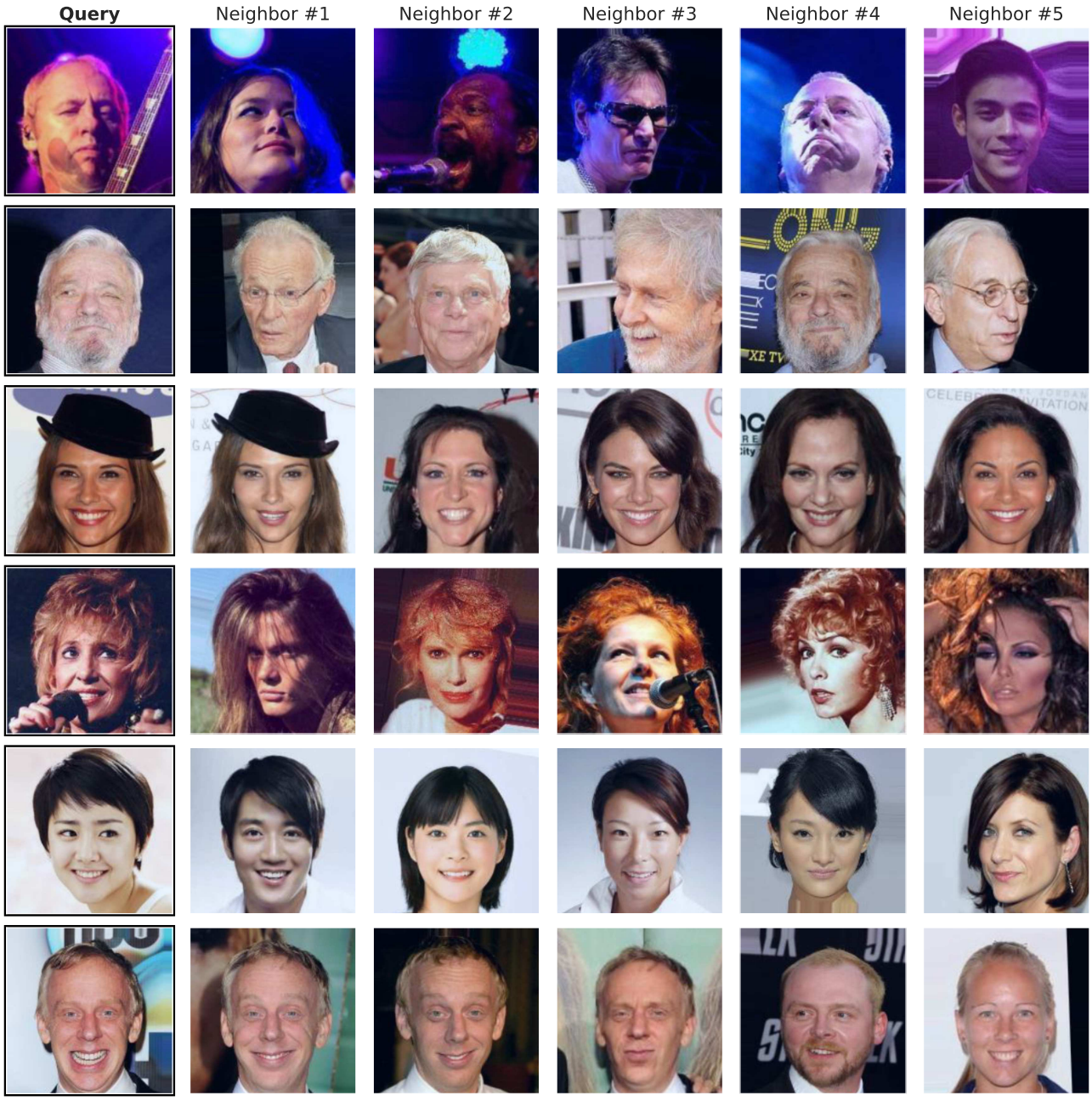

Figure 19: Visualizing nearest neighbors in COAR attribution space. We also use component attributions as feature embeddings in order to visualize the five nearest neighbors of examples from the CelebA dataset in "component attribution" space. Intuitively, this technique allows us to identify examples on which model outputs change similarly in response to component ablations. In general, we observe that the nearest neighbors of a given example in attribution space high visual similarity, e.g, similar facial attributes such as background (first row), hair color (second and fourth row), accessories (third row), or even the same person in different poses (last row).

H. Additional evaluation of COAR-EDIT

We use COAR-EDIT in five different editing tasks: correcting misclassifications (§5.1); forgetting a class (§D.1); improving subpopulation robustness (§5.2); localizing backdoor attacks (§D.2); and improving robustness to typographic attacks (§5.3). In this section, we provide additional details and/or supplementary experiments for each task.

H.1. Editing individual predictions

Experiment details. In Section 5.1, we use COAR-EDIT to correct misclassifications of a ResNet-50 model on ImageNet examples. In this experiment, we set the "target" example to be a misclassified ImageNet example and the "reference" example to a set of 50 randomly selected ImageNet examples. Then, we use these examples to identify and ablate components (9) that increase the correct-class margin (5) of the target example without impacting the average margin over the reference examples.

Additional experiments. We first show that COAR-EDIT is not sensitive to the choice of misclassified examples, model, or dataset. In Figure 21, we reproduce the experiment in Section 5.1 on additional ImageNet examples misclassified by a ResNet-50 model. In Figure 20, we use COAR-EDIT to similarly fix misclassifications of a ResNet-18 model on the CIFAR-10 dataset. In Figure 22, we show that COAR-EDIT can also be used to adversarially induce misclassifications on ImageNet examples by ablating the top-k components corresponding to the "target" example. Similar to our findings in Section 5.1, we observe that ablating a few components via COAR-EDIT is sufficient to change the individual example-level prediction without changing overall model performance.

Additional analysis. Which components does COAR-EDIT ablate to correct misclassifications? To answer this question, we first aggregate all components ablated by COAR-EDIT in order to (individually) correct ImageNet examples misclassified by a ResNet-50 model. Then, we plot the most common convolution layers corresponding to these ablated components in Figure 23. We find that COAR-EDIT primarily targets convolution filters from the last few layers (closet to the output) of the ResNet-50 model in order to make fine-grained edits that do not impact overall model performance. For example, more than 25% of the ablated components belong to `layer4.{0,1,2}.conv3`—the last convolution layer in the first three residual blocks of the last layer group of the ResNet-50 model.

H.2. Forgetting a class

*Experiment details. In Appendix D.1, we use COAR-EDIT to selectively forget a class of a ResNet-50 model on ImageNet. In this experiment, we set the "target" examples to be set of 10 examples from the class to be forgotten and the "reference" examples to be a set of 50 randomly selected ImageNet examples. Using these examples, we use COAR-EDIT to ablate components (9) that decrease the average correct-class margin (5) of the target examples without impacting the average margin over the reference examples.

*Additional experiments We show that COAR-EDIT can be used to selectively forget additional ImageNet classes. Specifically, in Figure 24, we reproduce the COAR-EDIT experiment in Appendix D.1 on three additional ImageNet classes: "folding chair", "military uniform", and "revolver". Like in Figure 6, we again observe that COAR-EDIT can specifically degrade the accuracy on the target class without impacting the average accuracy over the train or test set by ablating a few components (convolution filters) in the ResNet-50 model.

H.3. Improving subpopulation robustness.

*Experiment details. In Section 5.2, we use COAR-EDIT to improve subpopulation robustness of models trained on two benchmark datasets: Waterbirds and CelebA. In both cases, we fine-tune a ResNet-50 model via standard "empirical risk minimization" using SGD hyperparameters taken from Sagawa et al. (2020). The resulting fine-tuned models attain 64% and 47% worst-subpopulation accuracy on the Waterbirds and CelebA test sets, respectively. To improve subpopulation robustness on Waterbirds, we set the "target" examples to a set of 10 random training examples from the "waterbirds on land" (the worst-performing subpopulation) and the "reference" examples to be 10 random examples from other subpopulations. Analogously, for CelebA, we set the "target" examples to the set of 20 random examples from the "blond male" worst-performing subpopulation and the "reference" examples to 20 random examples from other subpopulations. Then, we use COAR-EDIT to identify components that, when ablated, increase the average correct-class margin (5) of the target examples without impacting the average margin over the reference examples. In both cases, the number of components to ablate is a hyperparameter that we select by tracking the worst-subpopulation accuracy on a validation set.

H.4. Mitigating backdoor attacks.

Experiment details. We now describe the experiment setup in Appendix D.2, where we used COAR-EDIT to mitigate the effect of a backdoor attack on a ResNet-18 model trained on a backdoored CIFAR-10 dataset. The CIFAR-10 dataset is modified by adding a small blue-squared trigger to the upper left corner of 50% of examples in the "airplane" class. Training a model with standard SGD hyperparameter on this dataset causes the model to spuriously associate the trigger with the "airplane" class, leading to a backdoor attack. That is, while the resulting model attains 89% test accuracy, applying the attack to examples in the test set causes the model to misclassify them as "airplanes", resulting in 37% accuracy on test examples with the trigger. To mitigate the effect of the backdoor attack, we first sample ten examples from the training set. Then, we set the "target" examples to these two examples *with* the trigger and the "reference" examples to these two examples *without* the trigger. Then, we use COAR-EDIT to ablate components (9) that increase the correct-class margin (5) of the target examples without impacting the average margin over the reference examples.

Additional analysis. Recall that our experiment in Appendix D.2 shows that COAR-EDIT can significantly mitigate the effect of a backdoor attack on a ResNet-18 model by ablating a few backdoor-specific components. We now qualitatively analyze the components ablated via COAR-EDIT to mitigate the effect of a backdoor attack in Figure 25. Specifically, we visualize the ablated components (convolution filters in this case) using the input-times-gradient saliency map method from the Captum library (Kokhlikyan et al., 2020). As shown in Figure 25, these visualizations suggest that the ablated components are sensitive to the blue-squared trigger.

H.5. Improving robustness to typographic attacks.

Experiment details. In Section 5.3 and Figure 5 in particular, we show that COAR-EDIT can be used to improve robustness of zero-shot CLIP classifiers to typographic attacks. In this experiment, we consider a zero-shot CLIP ViT-B/16 classifier (Radford et al., 2021) and specify a computation graph over $82,944$ components, where each component corresponds to a weight vector in the ViT (across all layers). We evaluate the robustness of this model in a zero-shot setting on 180 images and four real-world typographic attacks—"taxi", "twitter", "EU", and "iPad"—taken from the dataset in (Materzyńska et al., 2022). We also consider synthetic typographic attacks, where we render a blob of text on a white background and place it in the center of a given image. The zero-shot performance of the CLIP model drops from 95% to 51% and 54% on the real and synthetic typographic attacks, respectively. To improve robustness, we set the "target" examples to be the 25 examples *with* a randomly picked synthetic attack and the "reference" examples to the same set of examples *without* any attack. Then, we use COAR-EDIT to ablate components (9) that increase the average correct-class margin (5) of the target examples without impacting the average margin over the reference examples. We use a validation set comprising examples with and without the synthetic attack to select the number of components to ablate from the model.

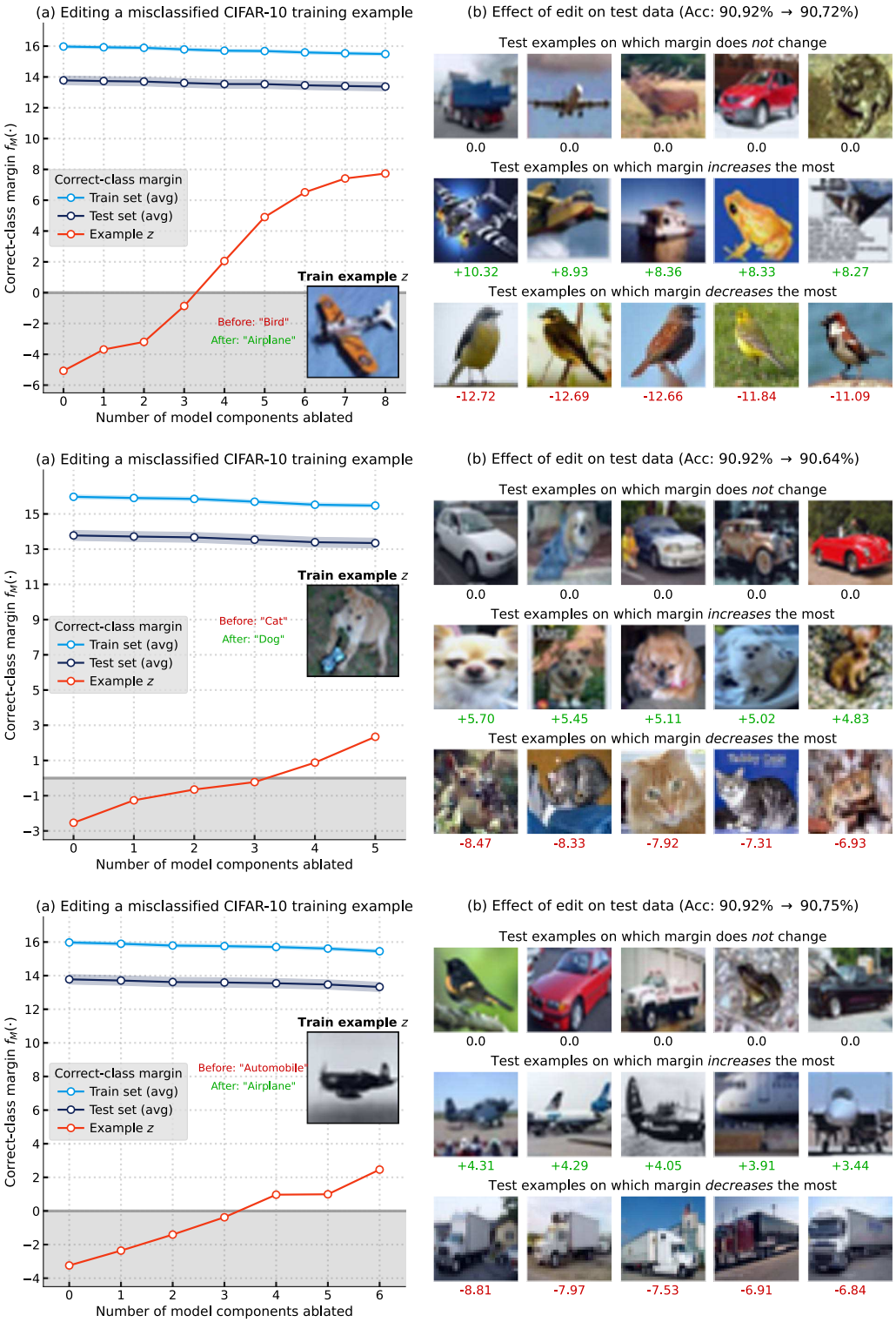

Figure 20: Correcting misclassified CIFAR-10 examples via COAR-EDIT. We reproduce the COAR-EDIT experiment from Section 5.1 on the CIFAR-10 dataset. Specifically, each row corresponds to CIFAR-10 test example that is misclassified by a ResNet-18 model. The left subplot in each row shows how applying COAR-EDIT (by ablating components (9)) increases the correct-class margin (5) of the misclassified example without impacting the average margin over the train or test set. The right subplot reports the drop in overall test accuracy and visualizes examples with correct-class margins that change the most or least due to the edit.

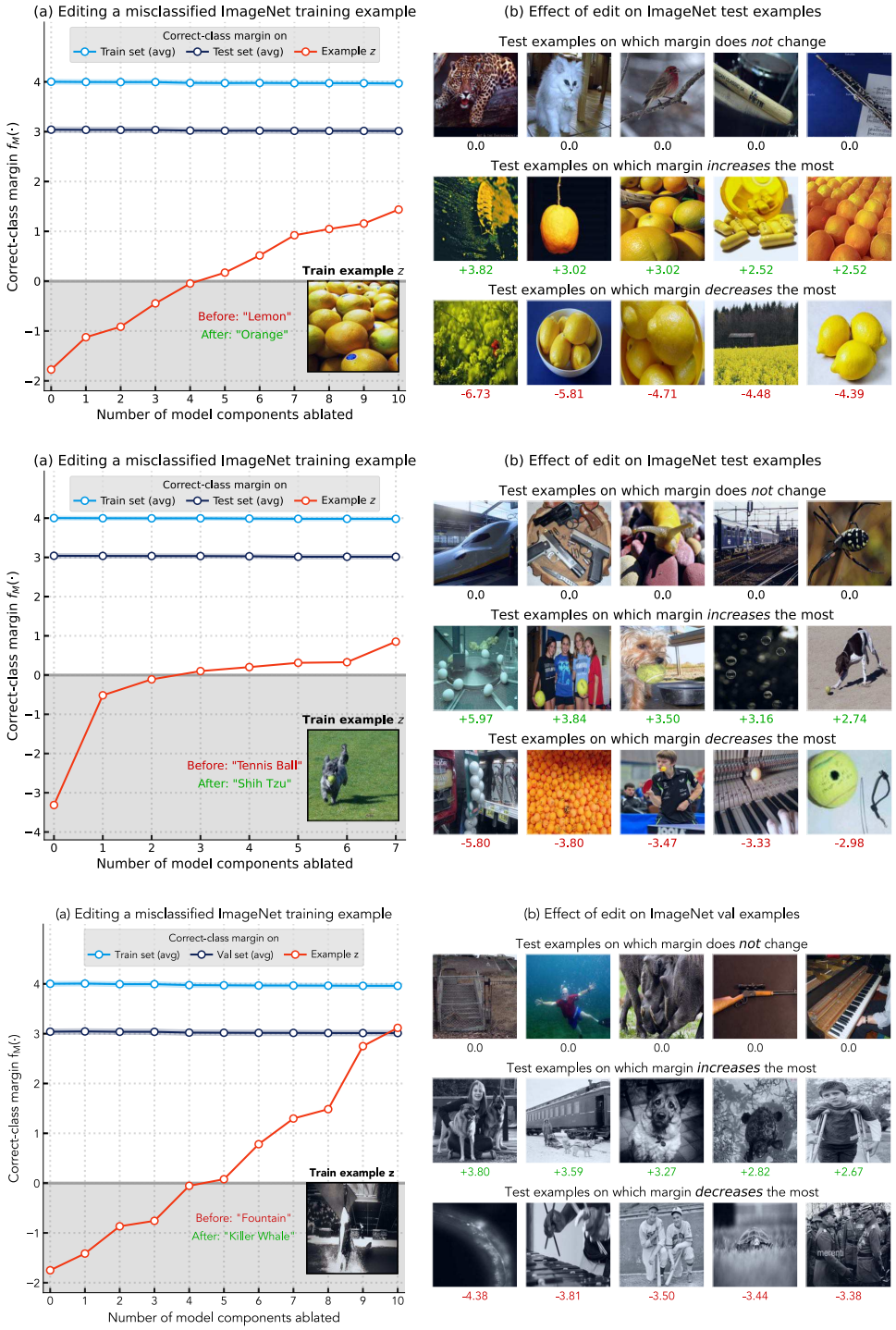

Figure 21: Correcting misclassified ImageNet examples via COAR-EDIT. We reproduce the COAR-EDIT experiment from Section 5.1 on additional ImageNet examples (one per row) misclassified by a ResNet-50 model. The left subplot shows that applying COAR-EDIT (by ablating components (9)) increases the correct-class margin (5) of the misclassified example without impacting the average margin over the train or test set. (Right) The right subplot visualizes examples with margins that change the most or least due to the edit.

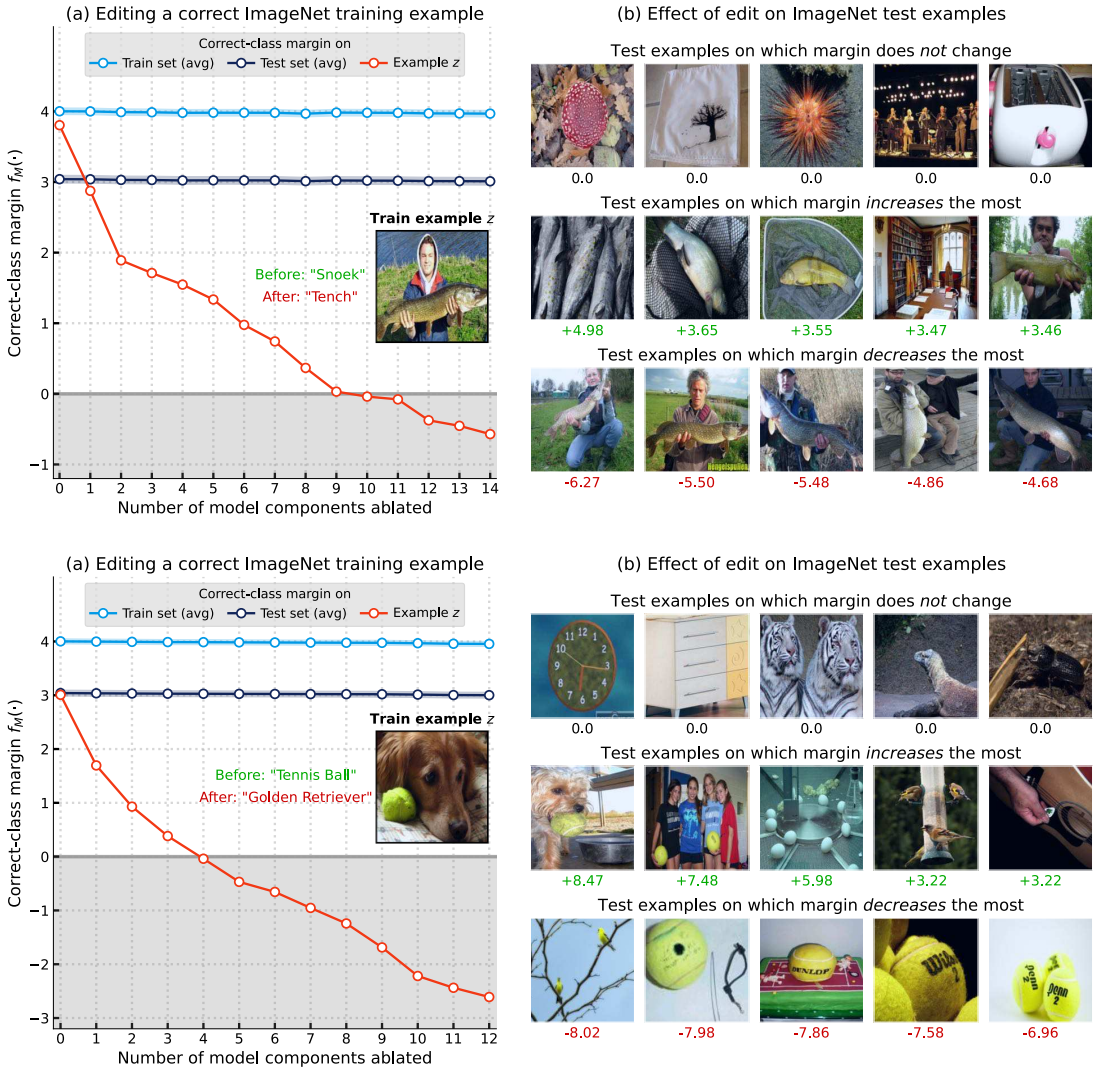

Figure 22: Adversarially inducing misclassifications on ImageNet examples via COAR-EDIT. Each row corresponds to an ImageNet test example that is correctly classified by a ResNet-50 model. In the left subplot of each row, we show that applying COAR-EDIT (by ablating the top-k components (9)) decreases the correct-class margin (5) of the correctly classified example without impacting the average margin over the train or test set. On the right, we shw that the edit does not impact visually dissimilar examples, but does increase or decrease the correct-class margin of examples containing visually similar objects, e.g., tennis balls in the second row.

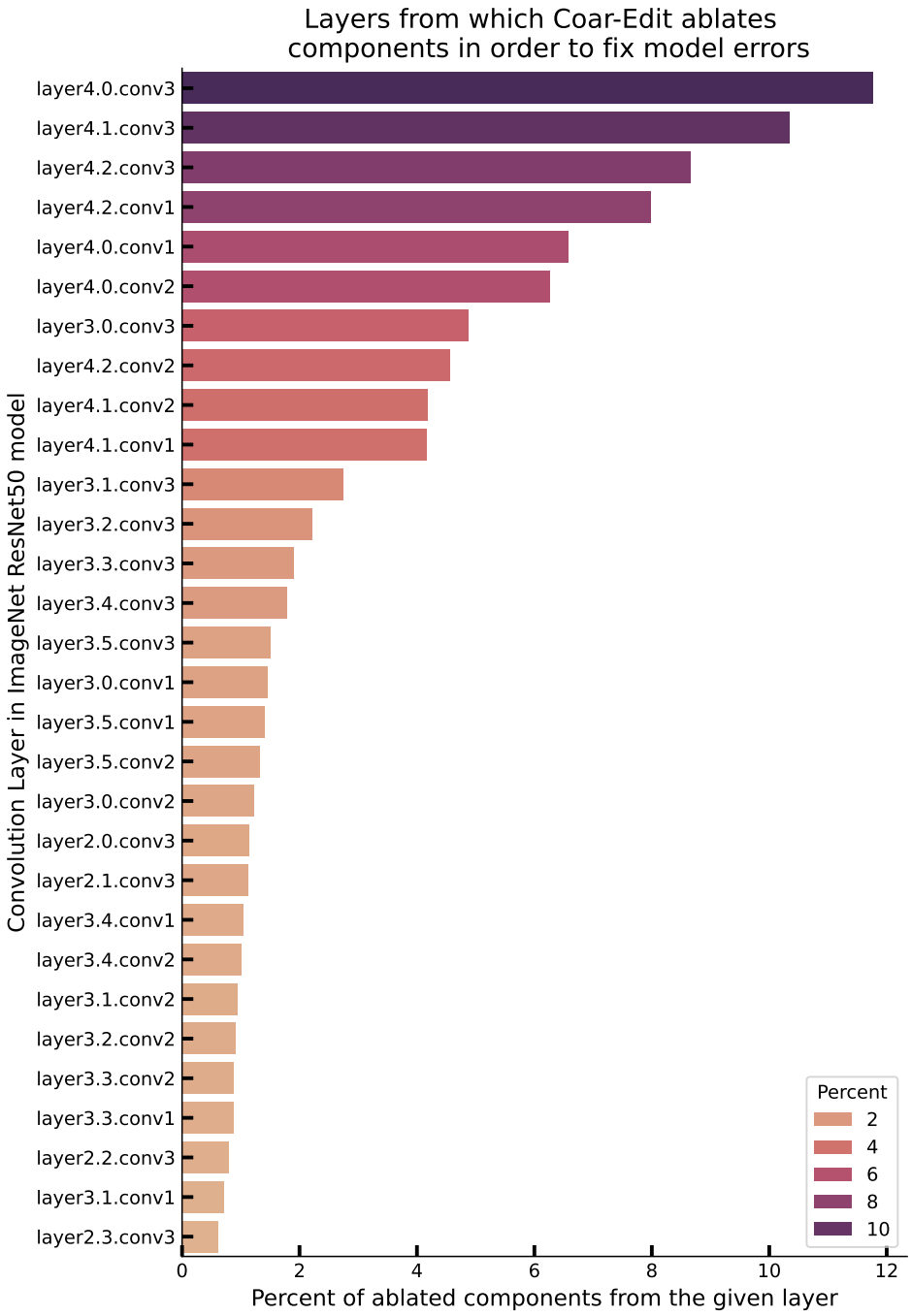

Figure 23: Which components does COAR-EDIT target to fix model errors? We analyze the specific convolution layers from which COAR-EDIT ablates components (convolution filters) to correct ImageNet examples misclassified by a ResNet-50 model. On the y-axis, we plot the 30 most common convolution layers corresponding to the ablated components. On the x-axis, we plot the percentage of ablated components that belong to each convolution layer. We find that COAR-EDIT primarily targets convolution filters from the last few layers (closet to the output) of the ResNet-50 model in order to make fine-grained edits that do not impact overall model performance. For example, more than 25% of the ablated components belong to `layer4.{0,1,2}.conv3`—the last convolution layer in the first three residual blocks of the last layer group of the ResNet-50 model.

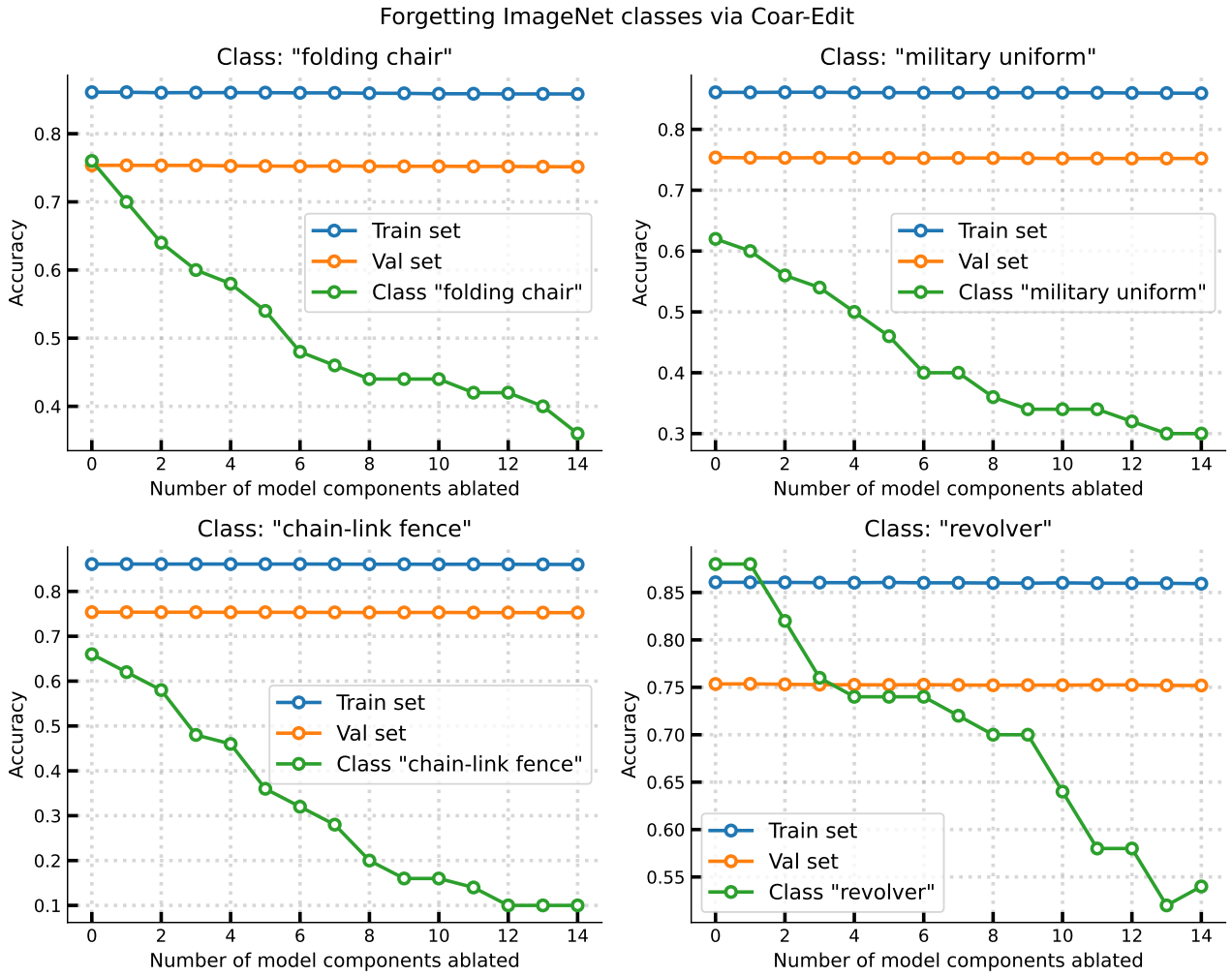

Figure 24: Forgetting ImageNet classes via COAR-EDIT. We reproduce the COAR-EDIT experiment from Appendix D.1 on additional ImageNet classes (one per subplot). Specifically, in each subplot, we find that ablating 15 of $22,720$ convolution filters (identified via COAR-EDIT) suffices to significantly degrade the accuracy of a ResNet-50 model on a specific class (in green). This edit is targeted in that it does not impact the average accuracy over the train set (in blue) or test set (in orange).

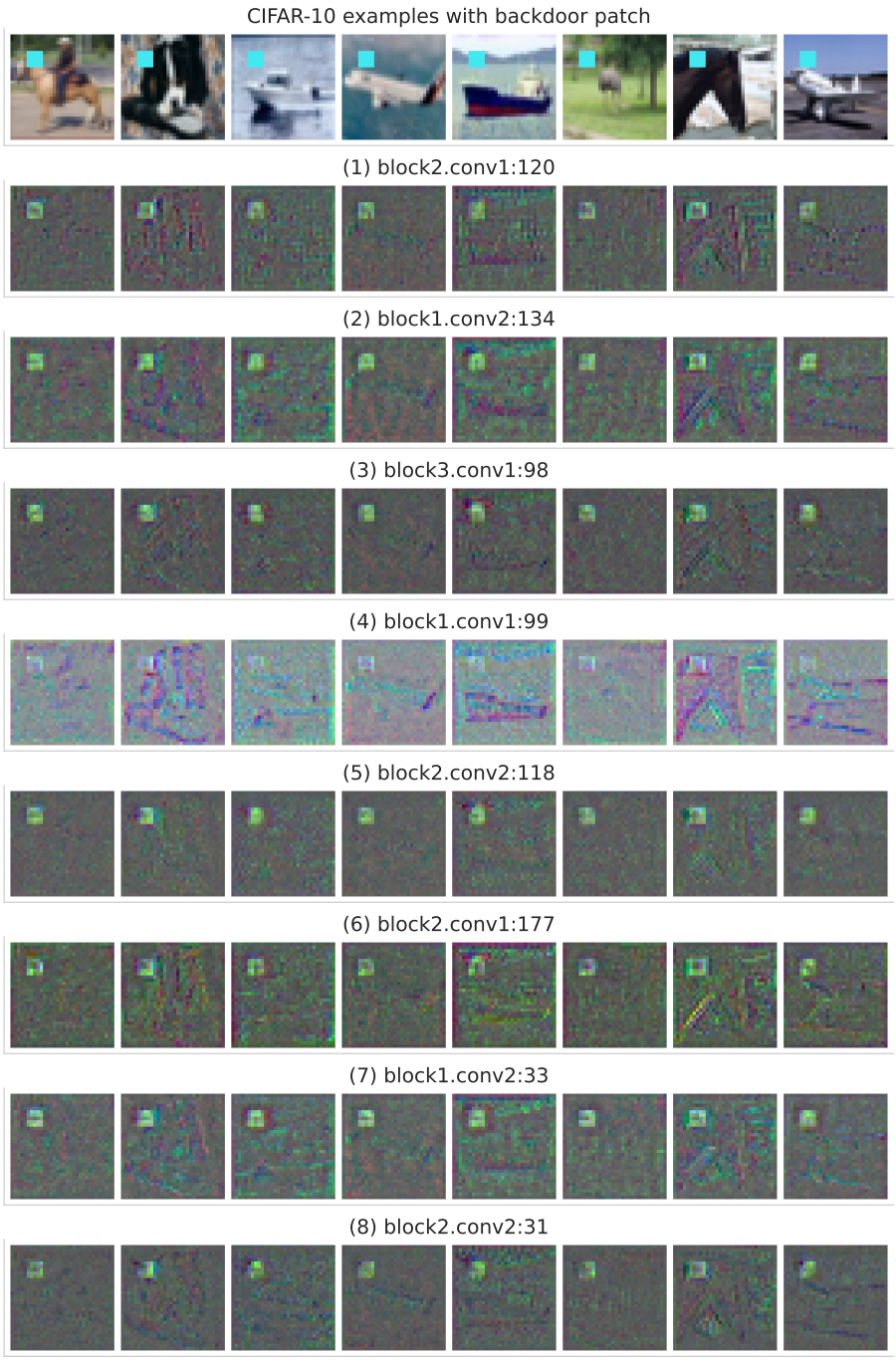

Figure 25: Visualizing components ablated via COAR-EDIT to mitigate a backdoor attack. Recall that in Appendix D.2, we used COAR-EDIT to mitigate the effect of a backdoor attack (a blue-squared spurious trigger) on a ResNet-18 model trained on a backdoored CIFAR-10 dataset. Here, we visualize the components ablated via COAR-EDIT to reduce the model's reliance on this spurious feature. The first row shows a set of random examples from the modified CIFAR-10 test set that contain the trigger. Each subsequent row corresponds to an ablated component—a convolution filter of the ResNet-18 model in this case. In each of these rows, we use the input-times-gradient saliency map method from the Captum library (Kokhlikyan et al., 2020) to (qualitatively) highlight parts of the examples that are most "important" for the ablated component's output. These maps suggest that all ablated components are sensitive to the blue-squared trigger.

I. Analyzing design choices in COAR

In this section, we analyze three design choices in COAR: (a) the train-time ablation fraction α used to sample a subset of components $C' \subset C$ of size $\alpha|C|$, (b) the ablation method (Remark 2) used to intervene on the sampled components C', and (c) the specific model output function used to compute component counterfactuals $f_M(\cdot, C')$ (1), i.e., model output $f_M(\cdot)$ after ablating the component subset C'.

I.1. Effect of ablation fraction

The first step of COAR—constructing a component dataset (Equation 2)—requires choosing a ablation fraction $\alpha \in (0, 1)$. This hyperparameter determines the size of the random α-fraction subsets $C' \subset C$ used to compute component counterfactuals. A priori, however, it is not clear which ablation fraction α is best suited for learning accurate component attributions. For example, ablating too large a component subset (large α) can induce a significant drop in model performance to a point where the ablated model is no longer representative of the original model.

*Effect of train-time ablation fraction α_{train} We use two metrics to quantify the effect of ablation fraction α on model outputs:

- **Change in model performance.** We measure the effect of ablating random α-fraction subsets $C' \subset C$ of components on model performance, e.g., test accuracy.
- **Correlation between example-level model outputs.** We measure the correlation between model outputs before and after ablation, e.g., logits or margins.

We use these (heuristic) metrics to ensure that the ablations are not too severe to nullify model performance and that the outputs of the ablated models are still predictive of the outputs of the original model.

*Effect of train-time ablation fraction α_{train}. Figure 26 evaluates how varying the train-time ablation fraction α_{train} changes both metrics—model performance and correlation between model outputs—for all three settings considered in Section 4: CIFAR-10 ResNet-18, ImageNet ResNet-50, and ImageNet ViT-B/16. In all three settings, we find that model accuracy and margin correlation decrease as the ablation fraction α increases. For instance, ablating 15% of components ($\alpha = 0.15$) results in a significant accuracy drop for ResNets, but not for ViTs. On the other hand, ablating 1% of all components ($\alpha = 0.01$) results in a small drop in accuracy and correlation, e.g., for the ResNet-18 model trained on CIFAR-10 (first row of Figure 26). Therefore, our experiments in Section 4 use $\alpha = 0.10$ for the CIFAR-10 model and $\alpha = 0.05$ for both ImageNet models. These findings also suggest that the choice of α depends on the model architecture and the task at hand, e.g., ViTs are more robust to zero ablations than ResNets.

I.2. Effect of ablation method

As discussed in Remark 2, we use a simple ablation method that sets the weights/activations of a subset of components $C' \subset C$ to zero. However, our method COAR is not dependent on any specific ablation method, and can be used to compute component attributions with other ablation methods as well.

Alternative ablation method based on scaling. In this section, we consider an alternative ablation method that scales down the activations of a component by a factor of $\gamma \in [0, 1]$. Note that setting $\gamma = 0$ corresponds to the zero ablation method described in Remark 2; we use $\gamma = 0.5$ in our experiments.

Experiment results. We find that the alternative scaling-based ablation maintains high correlation between model outputs before and after ablations, resulting in accurate component attributions. Specifically, we make three key observations.

- We first observe that on a ResNet-18 model trained on CIFAR-10, the scaling-based ablation method described above maintains high correlation between model outputs before and after ablation, even at high ablation fractions $\alpha \in \{0.30, \ldots, 0.05\}$ (fourth row of Figure 26).
- Then, in Figure 27, we apply COAR with the scaling-based ablation method to a CIFAR-10 ResNet-18 model. The resulting component attributions attain an average correlation of more than 0.75 for most ablation fractions $\alpha \in \{0.40, \ldots, 0.01\}$. The correlation between COAR attribution estimates and ground-truth counterfactuals is high across a range of ablation fractions α from 0.01 to 0.45.

- In Figure 28, we compare COAR attributions computed with the scaling ablations to attributions computed with zero-ablations. We find that (a) these attributions exhibit high cosine similarity (Figure 28a) and that (b) attributions learned with scaling-based ablations are predictive of ground-truth component counterfactuals computed using zero-ablations (Figure 28b). This indicates that both ablations—scaling down the activations of a component by a factor of $\gamma = 0.5$ and setting the activations of a component to zero—change model outputs in a similar manner.

I.3. Effect of model output function

Recall that in Section 4, we use the correct-class margin (5) as the model output function to estimate COAR attributions for classification tasks. However, our approach is not tied to a specific model output function. Depending on the task at hand, one can use an alternative model output function to estimate COAR attributions. For example, in a multi-label classification task, we can also use the logit of a fixed class of interest as the model output function to estimate COAR attributions. In Figure 13, we apply COAR to a pre-trained ImageNet ResNet50 model fine-tuned on MIMIC-CXR (Johnson et al., 2019)—a dataset of labeled chest radiographs—and set the model output function to be the logit of the "Cardiomegaly" class. Our results show that COAR attributions remain predictive with this model output function, and attain a correlation of 0.7 and 0.6 with the ground-truth counterfactuals on "Cardiomegaly" logits when $\alpha = \alpha_{\text{train}} = 0.05$ and $\alpha = 0.10$ respectively. Additionally, in Appendix F, we also apply COAR to the next-token prediction task in language modeling, using average correct-class margin over all tokens in a given sequence as the model output function.

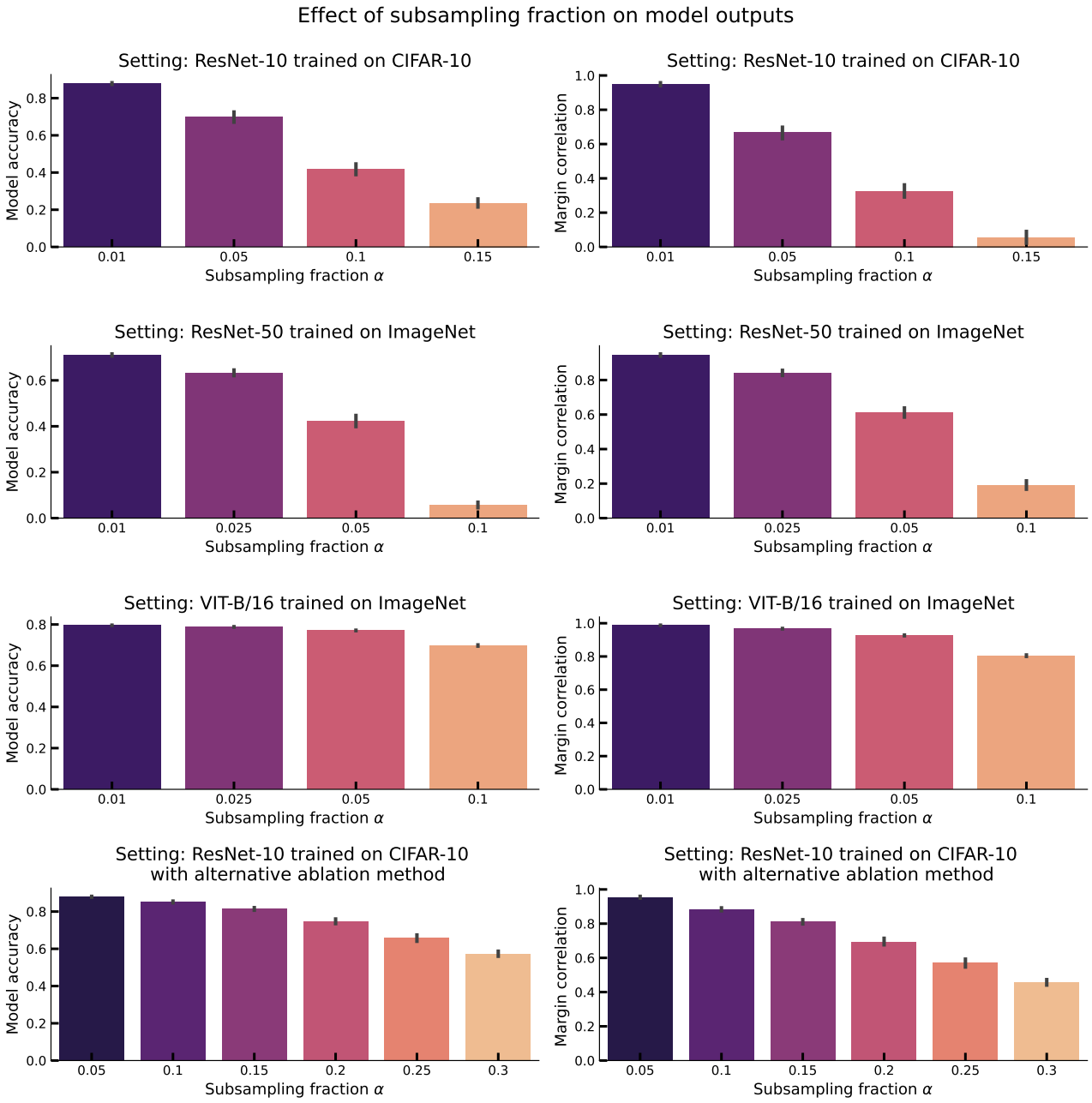

Figure 26: Effect of ablation fraction α on model outputs. We evaluate the effect of ablating α-fraction subsets $C' \subset C$ of components (x-axis) on model accuracy (y-axis in the left column) and the correlation between model outputs before and after ablation (y-axis in the right column). In all settings considered in Section 4 (one per row), we find that model accuracy and margin correlation gradually decrease as the ablation fraction α increases. See Appendix I.1 for more details.

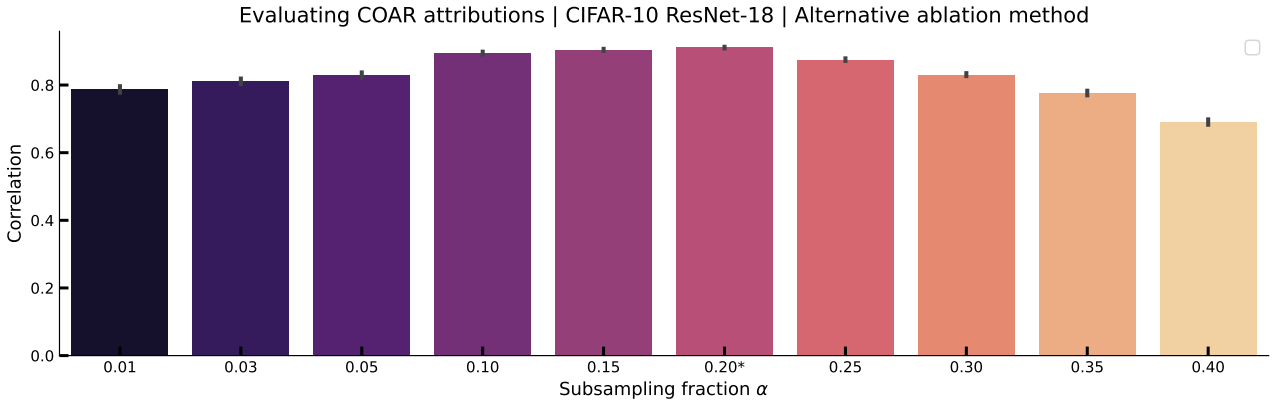

Figure 27: Effect of ablation method on COAR attributions. We estimate COAR attributions for a CIFAR-10 ResNet-18 model using an alternative ablation method that scales down the activations of a subset of components $C' \subset C$ by a factor of γ (0.5 in this case) instead of setting them to zero. The resulting attribution-derived estimates (4) exhibit high correlation (y-axis) with ground-truth component counterfactuals. See Appendix I.2 for more details.

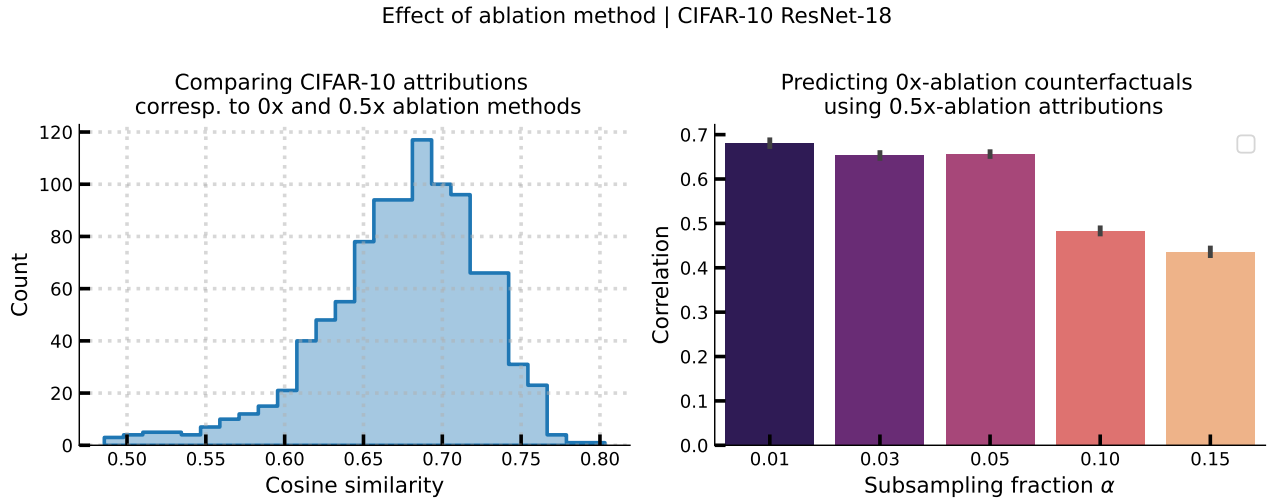

Figure 28: Comparing COAR attributions estimated with different ablation methods. We compare COAR attributions on a CIFAR-10 ResNet18 model computed with the zero-ablation method Remark 2 to attributions computed with the alternative ablation method described in Appendix I.2. The left plot shows that the paired attributions (corresponding to each example) exhibit high cosine similarity. The right plot shows that the counterfactual estimates (4) computed using attributions from the alternative ablation method are predictive of ground-truth component counterfactuals computed using the zero ablation method. See Appendix I.2 for more details.