Decomposing & Editing Predictions by Modeling Model Computation

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2024

Abstract

How does the internal computation of a machine learning model turn inputs into predictions? To tackle this question, we introduce *component modeling*, a framework for decomposing a prediction in terms of *model components*—architectural “building blocks” such as convolution filters or attention heads. We focus on a sub-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 estimator for component attribution, and showcase its effectiveness on vision and language models. Finally, we show that COAR directly enables effective model editing.

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 [9, 44, 101].

In response to this difficulty, a line of work in ML interpretability aims to shed light on model computation by analyzing *model components*—“grouped” model parameters such as convolutional filters or attention heads. Feature visualization methods [47, 102, 113] identify components in vision models that detect concepts such as curves [85] and objects [6]. Representation-based probes [1] identify language model components that encode sentiment [91], part-of-speech tags [13], and syntactic structure [56]. Mechanistic interpretability [81, 110] uncovers specific components that encode a model behavior of interest, e.g., “knowledge neurons” [26], “induction heads” [87]. Broadly, these works develop tools to answer: *How do individual 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 question into a concrete framework called *component modeling* for decomposing and editing predictions by intervening on model components. Specifically, we:

(1) Introduce the component modeling framework: We formalize our goal of understanding how model components shape predictions through a framework called *component modeling* (Def 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).

(2) Instantiate the framework via component attribution: We focus our attention on a special “linear” case of component modeling called *component attribution*, 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).

(3) Propose an estimator for component attribution: We develop COAR (component attribution via regression), a scalable estimator for component attributions (Section 3). Our findings show that COAR yields component attributions that can accurately predict how predictions of large-scale vision and language models change in response to component-level ablations (Section 4).
(4) Edit model behavior via component attribution: COAR directly enables zeroth-order model editing without any additional training. In Section 5, we outline COAR-EDIT, an editing method that ablates targeted component subsets to induce desired model behavior. We apply COAR-EDIT to five tasks: boosting subpopulation robustness (§ D.3), fixing model errors (§ D.1), "forgetting" classes (§ D.2), mitigating typographic attacks (§ D.4), and localizing backdoor attacks (§ D.5).

2. Setup and Problem Statement

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* $f_M(z) \in \mathbb{R}$ as any statistic that quantifies the correctness of model M on the example z e.g., cross-entropy loss in a classification task. We view the model M as a *computation graph* G_M [7], where each parameterized node—which we call a *component*—is a function mapping its incoming edges to an outgoing edge.

Component modeling. Viewing the model M as a computation graph G_M over components C, we can restate our goal as: *Given a model M and example z, how does every component $c \in C$ combine to output $f_M(z)$?* What we want is an *interpretable* function capturing how components in C impact $f_M(z)$. To make this precise, we define the *component counterfactual* $f_M(z, C')$ as

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

where "ablating" here corresponds to any intervention that patches the parameters corresponding to components $c \in C'$ (e.g., by zeroing out [87] or adding noise [77]). Eq. (1) allows us to operationalize our goal as the task of estimating component counterfactuals $f_M(z, C')$ using a much simpler surrogate function, which we call a *component model*.

Definition 1 (Component modeling) *Fix a model M with computation graph G_M, components $C = \{c_1, \ldots, c_N\}$, and model output function f_M. For any subset of model components $C' \subseteq C$, let $\mathbf{0}_{C'}$ be the ablation vector of C', a N-dimensional vector where $\mathbf{0}_{C'}[i] = 0$ if $c_i \in C'$ and $\mathbf{0}_{C'}[i] = 1$ otherwise. Given an example z, a component model 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 $f_M(z, C')$.*

The high-level goal of component modeling is to build an estimator that can simulate counterfactauls 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 graph G_M and ablate components in C'.

Component attribution. We focus on a subcase of component modeling—which we call *component attribution*—where the function $g^{(z)}$ is *linear* in its input.

Definition 2 (Component attribution) *Given a model M with model output f_M and component a $C = \{c_1, \ldots, c_N\}$, a component attribution for example z is a set of coefficients $\boldsymbol{\theta}^{(z)} := \{\boldsymbol{w}_1^{(z)}, \ldots, \boldsymbol{w}_N^{(z)}, b^{(z)}\}$ that parameterize a linear component model, i.e., $f_M(z; C') \approx g^{(z)}(\mathbf{0}_{C'}) := \mathbf{0}_{C'}^\top \boldsymbol{w}^{(z)} + b^{(z)}$.*

The 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 over scores corresponding to components in $C \backslash C'$. In doing so, the attributions decompose the output $f_M(z)$ into component-wise contributions $\boldsymbol{w}_i^{(z)}$.

3. Component attribution with COAR

In this section, we describe COAR (component attribution via regression), a general component attribution method for models ranging from random forests to deep networks. 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:
(Step 1) Construct a component dataset. We construct a dataset $D^{(z)}$ of component counterfactuals for the example z. where each "datapoint" consists of a component subset $C_i \subseteq C$ and its counterfactual $f_M(z, C_i)$ (see (1)). To compute the latter, we ablate the components in C_i and evaluate the model on example z. For simplicity, we choose the component subsets C_i to be random α_{train}-fraction subsets of the component set C, for a ablation fraction hyperparameter $\alpha_{\text{train}} > 0$. The output is a *component dataset* $D^{(z)} = \{(C_i, f_M(z, C_i))\}_{i=1}^{m}$ comprising pairs of component subsets and their counterfactuals. We study the effect of ablation fraction α_{train} on COAR in Appendix I.1.
(Step 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, we minimize the squared loss between the component counterfactuals and their attribution-based estimates by solving a *linear regression* problem:

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

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

$$g^{(z)}(\mathbf{0}_{C'}) := \mathbf{0}_{C'}^{\top} \boldsymbol{w}^{(z)} + b^{(z)}. \tag{3}$$

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

Instantiating COAR for classification. In this paper, we use COAR to analyze models evaluated on classification tasks, for which we use the standard *correct-class margin* [61] as the model output

$$f_M(z) := \text{ (logit for correct class)} - \text{(highest logit for incorrect class)}, \tag{4}$$

where the 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 [87, 110]. We consider alternative model output functions and ablation methods in Appendices I.3 and I.2.

4. Does COAR learn accurate attributions?

We now apply and evaluate COAR on image classification and language modeling tasks.

Datasets, models, and components. We apply COAR to compute component attributions in three setups: (A) ResNet-18 on CIFAR-10, (B) ResNet-50 on ImageNet, and (C) Vision Transformer (ViT-B/16) on ImageNet. The component set in setup {A,B,C} comprises $2,306$ convolutional filters, $22,720$ convolutional filters, and $82,944$ weight matrix rows. We defer details to Appendix E.2.

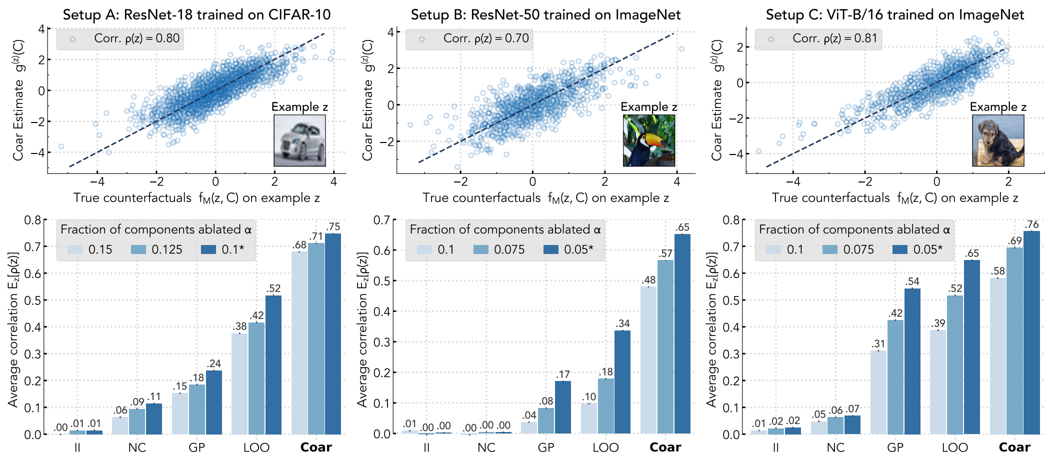


Figure 1: **Evaluating COAR.** We compare COAR to four baselines (described in Section 4) on three setups (one per column). In the first row, each subfigure focuses on a single example z (visualized in each plot), and shows that the component counterfactuals $f_M(z, \cdot)$ (x-axis) and attribution-based estimates $g^{(z)}(\cdot)$ (y-axis) exhibit high correlation $\rho(z)$. In the second row, we observe that COAR attributions exhibit high average correlation $\mathbb{E}_z[\rho(z)]$ over test examples, outperforming all baselines and for all ablation fractions α_{test}.

Evaluation metric. We evaluate the component attributions for example z based on its ability to estimate unseen component counterfactuals (1). Specifically, we sample a collection of k component subsets $D^{(z)} := C'_1, C'_2, \ldots, C_k$ where each $C'_i \sim \text{Unif}(C' \subset C : |C'| = \alpha_{\text{test}}|C|)$, with α_{test} as a hyperparameter. Setting $\alpha_{\text{test}} = \alpha_{\text{train}}$ (or not) evaluates the in-distribution (or out-of-distribution) performance of COAR. Using $D^{(z)}$, we compute the Pearson correlation $\rho(z)$ between component counterfactuals $f_M(z, C'_i)$ and their attribution-based estimates $g^{(z)}(\mathbf{0}_{C'_i})$:

$$\rho(z) := \rho_p\Big(\underbrace{\{f_M(z, C_1), \ldots, f_M(z, C_k)\}}_{\text{ground-truth counterfactuals}}, \underbrace{\{g^{(z)}(\mathbf{0}_{C'_1}), \ldots, g^{(z)}(\mathbf{0}_{C'_k})\}}_{\text{component attribution estimates}}\Big) \tag{5}$$

Baselines. We compare COAR to four baselines. The first two are adapted from related work: internal influence (`II`) [70] and neuron conductance (`NC`) [31]. The other two are natural baselines: leave-one-out (`LOO`) and gradient-times-parameter (`GP`). We provide details to Appendix E.3.

Results. The first row of Figure 1 corresponds to an individual example from each setup. We observe that COAR learns accurate component attributions that obtain high Pearson correlations $\rho(z)$ with the true counterfactuals. For each task, the second row of Figure 1 compares COAR to the baselines using the average correlation $E_z[\rho(z)]$ (5) averaged over all test examples. Our method COAR consistently outperforms all four baselines for multiple values of α_{test} across all setups.

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 yields accurate component attributions for language models: GPT-2 [92] evaluated on TinyStories [33] and Phi-2 [62] evaluated on BoolQ [23].

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?

The problem of component attribution and model editing are closely related. 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 to "*which components, when ablated, would change model outputs in a specific way?*" So, we use attributions to identify components that, when ablated, edit model behavior in a targeted manner. Specifically, we introduce COAR-EDIT, an editing approach based on COAR attributions. Given a model M with components C, target examples S_T sampled from $\mathcal{D}_T$, and reference examples S_R from $\mathcal{D}_R$, COAR-EDIT identifies a model edit in three steps:

1. Compute COAR attributions $\boldsymbol{\theta}^{(z)} = (\boldsymbol{w}^{(z)}, b^{(z)})$ for all target & reference examples $z \in S_T \cup S_R$.
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:

$$\tau(c_i) := \frac{\mu(S_T) - \mu(S_R)}{\sqrt{\frac{\sigma^2(S_T)}{|S_T|} + \frac{\sigma^2(S_R)}{|S_R|}}}, \text{ where } \begin{cases} \mu(S) = \frac{1}{|S|}\sum_{z \in S} w_i^{(z)} \\ \sigma^2(S) = \frac{1}{|S|}\sum_{z \in S}(w_i^{(z)} - \mu(S))^2. \end{cases} \tag{6}$$

3. Increase the model outputs on target examples S_T by ablating a set of components $C_{\text{edit}} = \arg \text{bottom-}k(\{\tau(c_i) : c_i \in C\})$ comprising the k most negative scores τ_i. The hyperparameter k can be tuned by cross-validation. To *decrease* outputs, we can replace bottom-k with top-k.

The score $\tau(c_i)$ in Equation 6 is a t-test statistic with the null hypothesis that component c_i has an equal average effect on the target and reference distributions. Using these scores, we find components that, if ablated, change target outputs the most relative to the change in reference outputs.

Applications. In Appendix D, we stress-test COAR-EDIT on five model editing tasks: fixing model errors (§D.1), "forgetting" specific classes (§D.2), boosting subpopulation robustness (§D.3), localizing backdoor attacks (§D.5), and improving robustness to typographic attacks (§D.4).

6. Related work

We defer details on related work, limitations, and future work to Appendices A, B, and C.

Localizing model behavior. COAR connects to work in mechanistic interpretability on identifying "subnetworks" within neural networks responsible for specific behaviors [18, 86, 87, 107, 110], and in particular finding these subnetworks in an automated manner [8, 25]. While recent work shows that these methods can be sensitive to design choices [114] and lack actionable insights for model editing [52], we show that COAR 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 [5, 10, 60, 98]. In Section 5, we show that COAR can enable editing by simply zeroing out specific components.

7. Conclusion

We introduce *component modeling*, a framework for decomposing predictions in terms of model components. We focus on the *component attribution* sub-case, where the goal is to predict the counterfactual impact of component ablations on a prediction. Our method, COAR, yields predictive attributions for large-scale vision and language models, which can directly inform model editing.

References

[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[2] Omer Antverg and Yonatan Belinkov. On the pitfalls of analyzing individual neurons in language models. *arXiv preprint arXiv:2110.07483*, 2021.

[3] Samyadeep Basu, Nanxuan Zhao, Vlad Morariu, Soheil Feizi, and Varun Manjunatha. Localizing and editing knowledge in text-to-image generative models. *arXiv preprint arXiv:2310.13730*, 2023.

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *European Conference on Computer Vision (ECCV)*, 2020.

[6] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences (PNAS)*, 2020.

[7] Friedrich L Bauer. Computational graphs and rounding error. In *SIAM Journal on Numerical Analysis*, volume 11, pages 87–96. SIAM, 1974.

[8] Deniz Bayazit, Negar Foroutan, Zeming Chen, Gail Weiss, and Antoine Bosselut. Discovering knowledge-critical subnetworks in pretrained language models. *arXiv preprint arXiv:2310.03084*, 2023.

[9] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *European Conference on Computer Vision (ECCV)*, 2018.

[10] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *arXiv preprint arXiv:2306.03819*, 2023.

[11] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.

[12] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 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.

[13] Terra Blevins, Omer Levy, and Luke Zettlemoyer. Deep rnns encode soft hierarchical syntax. *arXiv preprint arXiv:1805.04218*, 2018.

[14] Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

[15] Davis Brown, Charles Godfrey, Cody A. Nizinski, Jonathan Tu, and Henry Kvinge. Robustness of edited neural networks. *ArXiv*, abs/2303.00046, 2023.

[16] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency (FAccT)*, 2018.

[17] Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 5(6):e00024–003, 2020.

[18] Steven Cao, Victor Sanh, and Alexander M Rush. Low-complexity probing via finding subnetworks. *arXiv preprint arXiv:2104.03514*, 2021.

[19] Stephen Casper, Tilman Rauker, Anson Ho, and Dylan Hadfield-Menell. 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.

[20] Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. 2022.

[21] Ting-Yun Chang, Jesse Thomason, and Robin Jia. Do localization methods actually localize memorized data in llms? *arXiv preprint arXiv:2311.09060*, 2023.

[22] Haozhe Chen, Junfeng Yang, Carl Vondrick, and Chengzhi Mao. Interpreting and controlling vision foundation models via text explanations. *arXiv preprint arXiv:2310.10591*, 2023.

[23] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[24] Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *ArXiv*, abs/2307.12976, 2023.

[25] Arthur Conmy, Augustine N Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023.

[26] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

[27] Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 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, pages 6309–6317, 2019.

[28] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

[29] Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. Sparse interventions in language models with differentiable masking. *arXiv preprint arXiv:2112.06837*, 2021.

[30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[31] Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron? *arXiv preprint arXiv:1805.12233*, 2018.

[32] Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. Analyzing individual neurons in pre-trained language models. *arXiv preprint arXiv:2010.02695*, 2020.

[33] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[34] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[35] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 2881–2891, 2020.

[36] Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. *arXiv preprint arXiv:2106.06087*, 2021.

[37] Dan Friedman, Andrew Lampinen, Lucas Dixon, Danqi Chen, and Asma Ghandeharioun. Interpretability illusions in the generalization of simplified models. *arXiv preprint arXiv:2312.03656*, 2023.

[38] Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting clip's image representation via text-based decomposition. *arXiv preprint arXiv:2310.05916*, 2023.

[39] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023.

[40] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

[41] Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586, 2021.

[42] Atticus Geiger, Chris Potts, and Thomas Icard. Causal abstraction for faithful model interpretation. *arXiv preprint arXiv:2301.04709*, 2023.

[43] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019.

[44] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. In *Nature Machine Intelligence*, 2020.

[45] Robert Geirhos, Roland S Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. Don't trust your eyes: on the (un) reliability of feature visualizations. *arXiv preprint arXiv:2306.04719*, 2023.

[46] Kristian Georgiev, Joshua Vendrow, Hadi Salman, Sung Min Park, and Aleksander Madry. The journey, not the destination: How data guides diffusion models. *arXiv preprint arXiv:2312.06205*, 2023.

[47] Amin Ghiasi, Hamid Kazemi, Eitan Borgnia, Steven Reich, Manli Shu, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. What do vision transformers learn? a visual exploration. *arXiv preprint arXiv:2212.06727*, 2022.

[48] Amirata Ghorbani and James Y Zou. Neuron shapley: Discovering the responsible neurons. *Advances in neural information processing systems*, 33:5922–5932, 2020.

[49] Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021.

[50] Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

[51] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[52] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*, 2023.

[53] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. In *International Conference on Learning Representations (ICLR)*, 2019.

[54] Katherine L Hermann, Hossein Mobahi, Thomas Fel, and Michael C Mozer. On the foundations of shortcut learning. *arXiv preprint arXiv:2310.16228*, 2023.

[55] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*, 2021.

[56] John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368*, 2019.

[57] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*, 2018.

[58] Jing Huang, Atticus Geiger, Karel D'Oosterlinck, Zhengxuan Wu, and Christopher Potts. Rigorously assessing natural language explanations of neurons. *arXiv preprint arXiv:2309.10312*, 2023.

[59] Badr Youbi Idrissi, Martin Arjovsky, Mohammad Pezeshki, and David Lopez-Paz. Simple data balancing achieves competitive worst-group-accuracy. In *Conference on Causal Learning and Reasoning*, pages 336–351. PMLR, 2022.

[60] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[61] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *International Conference on Machine Learning (ICML)*, 2022.

[62] Mojan Javaheripi and Sébastien Bubeck. Phi-2: The surprising power of small language models, Dec 2023. URL https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/.

[63] Alistair EW Johnson, Tom J Pollard, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Yifan Peng, Zhiyong Lu, Roger G Mark, Seth J Berkowitz, and Steven Horng. Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042*, 2019.

[64] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

[65] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[66] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[67] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Technical report*, 2009.

[68] Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. The emergence of number and syntax units in lstm language models. *arXiv preprint arXiv:1903.07435*, 2019.

[69] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. ffcv. https://github.com/libffcv/ffcv/, 2022.

[70] Klas Leino, Shayak Sen, Anupam Datta, Matt Fredrikson, and Linyi Li. Influence-directed explanations for deep convolutional networks. In *2018 IEEE international test conference (ITC)*, pages 1–8. IEEE, 2018.

[71] Maximilian Li, Xander Davies, and Max Nadeau. Circuit breaking: Removing model behaviors with targeted ablation. 2023.

[72] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

[73] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3207–3216, 2020.

[74] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.

[75] Pratyush Maini, Michael C Mozer, Hanie Sedghi, Zachary C Lipton, J Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? In *International Conference on Machine Learning*, 2023.

[76] Joanna Materzyńska, Antonio Torralba, and David Bau. Disentangling visual and written concepts in clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16410–16419, 2022.

[77] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36, 2022.

[78] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

[79] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

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

[81] Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

[82] Lauren Oakden-Rayner, Jared Dunnmon, Gustavo Carneiro, and Christopher Ré. Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. In *Proceedings of the ACM conference on health, inference, and learning*, 2020.

[83] Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks. *arXiv preprint arXiv:2204.10965*, 2022.

[84] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. In *Distill*, 2018.

[85] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in inceptionv1. *Distill*, 2020. doi: 10.23915/distill.00024.002. https://distill.pub/2020/circuits/early-vision.

[86] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

[87] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[88] Vedant Palit, Rohan Pandey, Aryaman Arora, and Paul Pu Liang. Towards vision-language mechanistic interpretability: A causal tracing tool for blip. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2856–2861, 2023.

[89] Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*, 2023.

[90] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *Arxiv preprint arXiv:2303.14186*, 2023.

[91] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

[92] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[93] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *arXiv preprint arXiv:2103.00020*, 2021.

[94] Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan D Cotterell. Linear adversarial concept erasure. In *International Conference on Machine Learning*, pages 18400–18421. PMLR, 2022.

[95] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[96] Elan Rosenfeld and Andrej Risteski. Outliers with opposing signals have an outsized effect on neural network optimization. *arXiv preprint arXiv:2311.04163*, 2023.

[97] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2020.

[98] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In *Preprint*, 2021.

[99] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

[100] Harshay Shah, Prateek Jain, and Praneeth Netrapalli. Do input gradients highlight discriminative features? *Advances in Neural Information Processing Systems*, 34, 2021.

[101] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*, 2019.

[102] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[103] Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. Understanding arithmetic reasoning in language models using causal mediation analysis. *arXiv preprint arXiv:2305.15054*, 2023.

[104] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

[105] Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

[106] Joshua Vendrow, Saachi Jain, Logan Engstrom, and Aleksander Madry. Dataset interfaces: Diagnosing model failures using controllable counterfactual generation. *arXiv preprint arXiv:2302.07865*, 2023.

[107] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

[108] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[109] Haohan Wang, Songwei Ge, Eric P Xing, and Zachary C Lipton. Learning robust global representations by penalizing local predictive power. *Neural Information Processing Systems (NeurIPS)*, 2019.

[110] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. URL https://arxiv.org/abs/2211.00593.

[111] Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. A comprehensive survey of forgetting in deep learning beyond continual learning. *arXiv preprint arXiv:2307.09218*, 2023.

[112] Kaiyue Wen, Yuchen Li, Bingbin Liu, and Andrej Risteski. Transformers are uninterpretable with myopic methods: a case study with bounded dyck grammars. *arXiv preprint arXiv:2312.01429*, 2023.

[113] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[114] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

[115] Xiaosen Zheng, Tianyu Pang, Chao Du, Jing Jiang, and Min Lin. Intriguing properties of data attribution on diffusion models. *arXiv preprint arXiv:2311.00500*, 2023.

[116] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. In *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[117] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*, 2018.

[118] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*, 2020.

[119] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Appendices

Appendix 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" [86]. For example, prior work has used a variety of methods to localize gender bias [107]; specifical factual associations [77]; and other behavior [50, 71, 105, 110] within model parameters.

More recent work has tried to automate this localization process, using techniques based on fine-tuning [89], activation patching [25, 50], or differentiable masking [8, 18, 21, 29]. These techniques (or variants thereof) have been subsequently used to localize properties such as arithmetic reasoning [103], syntactic agreement [36], visual question answering [88], and visual attributes in diffusion models [3]. Other work has also developed methods [31, 48, 70, 80] to attribute model behavior to specific components.

Recently, however, Zhang and Nanda [114] 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. [52] 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 (**??**) 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" [28, 78], rank-one updates to model parameters [5, 77, 98], constrained fine-tuning [118], and weight interpolation [60, 119], among other methods. Recent work has also studied erasing concepts and suppressing spurious correlations from models using layer-wise linear probing [10, 94], CLIP-specific text-based methods [22, 38], and fine-tuning variants [39, 64]. In this work, we treat model editing as an application, and show how attributions can enable model edits that modify individual model predictions (§D.1,§D.2), improve subpopulation-level robustness (§D.3) and suppress spurious concepts (§D.5, §D.4).

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 [47, 85, 113], activation maps [4, 79], ablations [117], saliency maps [84], probing [27, 32], and natural language descriptions [12, 55, 83]. Subsequent analyses use these methods to identify and ascribe meaning to specific model components by labeling them as, e.g., "curve detectors" [17], "knowledge neurons" [26], "multimodal neurons" [49], and "syntax units" [68] to name a few. Recently, however, the reliability and robustness of these methods has been called into question [2, 14, 45, 56–58, 100]. 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 [95] approximate a given ML model with a linear model in input space. Similarly, a line of work on datamodeling [61, 90] 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* [41, 42] 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.

Appendix 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., Casper et al. [19], Meng et al. [77], Vig et al. [107], Wang et al. [110]) 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 [37, 112] 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. [52] 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 [24] and degrade robustness to distribution shifts [15]. On the other hand, recent work shows that localization methods can in fact recover "ground truth" localization in controlled settings [21] and improve calibration of fine-tuned language models [89]. 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 (**??**), more sophisticated ablations (e.g., Chan et al. [20]) may be more appropriate for different tasks and/or model architectures [114]. 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.

Appendix 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 [46, 115]. For language models, a possible point of start (following Park et al. [90]) would be to use the average correct-class margin (4) 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 [92] and Phi-2 [62] 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 Appendix D.1, can we use component attribution to better isolate "conflicting features" [61, 96] for a given task, and to understand their role in the training process [96]? 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 [35, 75]? Given that component attributions are causally meaningful, can we use them as a kernel with which to compare different models [66] or learning algorithms [**?**]? Relatedly, are component models transferable across tasks (allowing us to view them as sparse "subpopulation vectors" [60])?

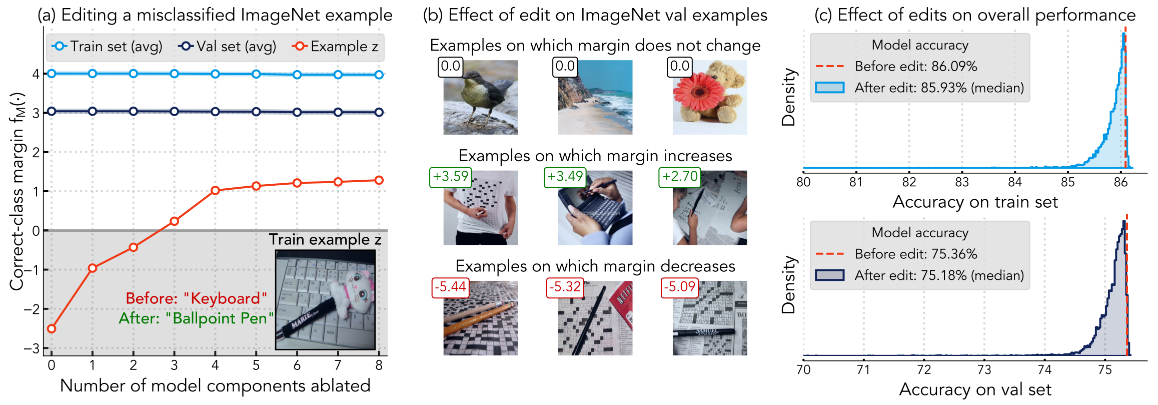


Figure 2: **Editing individual model predictions with COAR-EDIT.** We edit a ResNet50 model to correct a misclassified ImageNet example z shown on the left. Ablating a few components via COAR-EDIT (see (**??**)) increases the correct-class margin (4) on example z (red) without changing the average margin 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).

Appendix D. Additional COAR-EDIT experiments

D.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 (**??**) here would increase the model margin (4) 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 2. Figure 2a 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 2b then visualizes (again, for the specific example being edited in Figure 2a) the examples on which model outputs changes most (and least) drastically. Finally, Figure 2c 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.

D.2. "Forgetting" a class

We now consider "selective forgetting" problem [111], 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 Appendix D.1, with the goal of forgetting the entire "chain-link fence" class. Like before, we use our editing approach COAR-EDIT (see (6) and (**??**)) 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 3 summarizes our findings. In Figure 3a, 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 3b, 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 3c uses the ImageNet-Sketch [109] (top) and ImageNet$\star$ [106] (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.

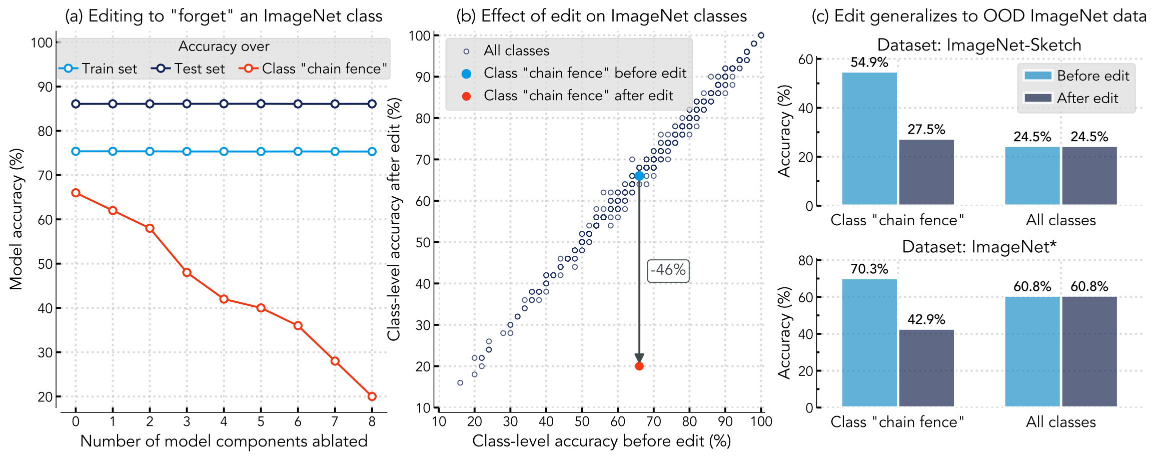


Figure 3: *"Forgetting" a class with* COAR-EDIT. We edit an ImageNet-trained ResNet-50 (Setup B from Section 4) to forget the "chain-link fence" class. On the left, we show how model accuracy on the class of interest (red) degrades as a function of k (the number of components removed), while model accuracy 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 [109] and ImageNet$\star$ [106]). As desired, our edit has a significant effect on the chain-link fence class, while leaving average model performance unchanged.

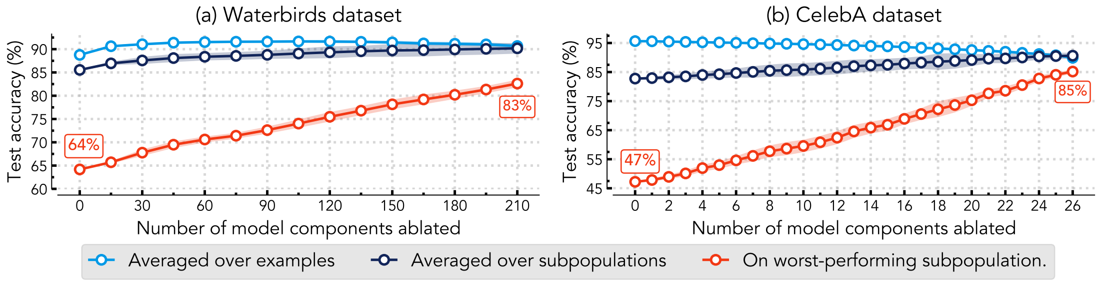


Figure 4: **Improving subpopulation robustness with Coar-Edit.** We edit pre-trained ResNet-50 models to improve their worst-subpopulation accuracy on Waterbirds [97] and CelebA [74]. Before editing, Waterbirds and CelebA models attain 87% and 96% test accuracy but only 64% and 47% accuracy on their 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%.

D.3. Improving subpopulation robustness

Machine learning models often latch onto spurious correlations in the training dataset [43, 54, 99], resulting in subpar performance on subpopulations where these correlations do not hold [16, 82]. 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 [97] and CelebA [74]—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 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 [97], dataset selection [59])

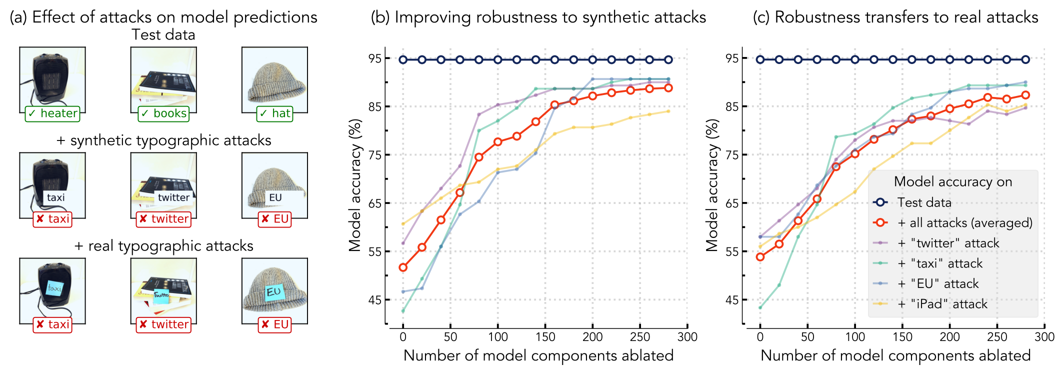


Figure 5: **Improving robustness to typographic attacks with COAR-EDIT.** We edit a zero-shot CLIP ViT-B/16 classifier to improve its robustness to typographic attacks [49]. On the left, we find that predictions on images of household objects (top row) can be manipulated to "taxi", "twitter", or "EU" 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.

D.4. Improving robustness to typographic attacks

Zero-shot CLIP classifiers [93] are vulnerable to typographic attacks [49] 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 [76] 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.

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.2) and localizing backdoor attacks (§D.5).

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.

D.5. Mitigating a backdoor attack

We now use COAR to edit a model in order to reduce its sensitivity to backdoor attacks [11, 51]. 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. [51] 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 6a). As shown in Figure 6a, 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 6b 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 6c, 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.

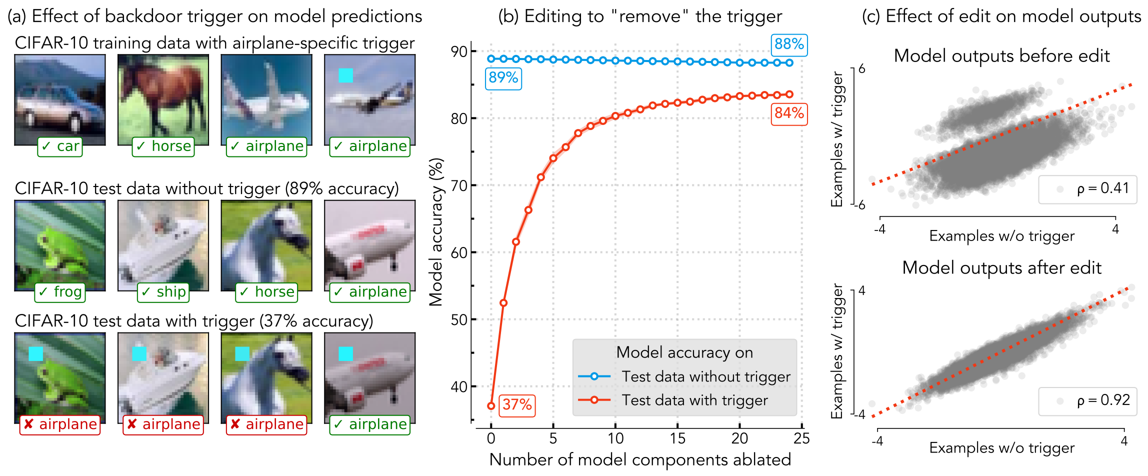


Figure 6: *Mitigating backdoor attacks with* COAR-EDIT. We edit a ResNet18 trained on a backdoored CIFAR-10 dataset to remove its dependence on a planted blue-squared trigger that is spuriously correlated with the “airplane” class. On the left, we show that the model is sensitive to the trigger—adding 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.

Appendix 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.

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

Figure 7: Pseudocode for estimating component attributions with Coar.

E.2. Datasets, models, components, and applying Coar.

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

CIFAR-10. We use the standard CIFAR-10 [67] image classification dataset to evaluate Coar attributions (Section 4, Appendix G.2) and for an editing task (Appendix D.5). 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 [30] image classification dataset to evaluate Coar attributions in Section 4 and for editing tasks in Appendix D.2. We use ImageNet-Sketch [109] and five random shifts from ImageNet⋆ [106]—“in the water”, “at dusk simple”, “orange”, “pencil sketch”, “green”— to evaluate the out-of-distribution performance of edited ImageNet models in Appendix D.2. We use the pre-trained ResNet50 and ViT-B/16 models[1] 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 [97] comprises images of birds taken from the CUB dataset [108] and pasted on backgrounds from the Places dataset [116]. The task here is to classify “waterbirds” and “landbirds” in the presence of spurious correlated “land” and “water” backgrounds in the

1. Model and pre-trained weights taken from torchvision: `https://pytorch.org/vision/stable/models.html`

training dataset. Sagawa et al. [97] 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 [73] comprises images of celebrities with binary attributes such as "smiling", "wearing hat", "wearing lipstick", etc. Similar to previous work on subpopulation robustness (e.g., [97]), 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, each corresponding to a convolution filter.

Typographic attacks dataset. We use a dataset of typographic attacks [76] for an editing task in Appendix D.4. 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 Appendix D.4 uses this dataset along with a zero-shot CLIP ViT-B/16 classifier [93]. 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 [33] 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 [23] to evaluate COAR attributions for the Phi-2 model [72]. 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. [40][2]. 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.

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 (4) 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.

2. https://github.com/EleutherAI/lm-evaluation-harness/

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 $\boldsymbol{w}^{(\boldsymbol{z})} \in \mathbb{R}^{|C|}$ for a given example z by assigning an "importance" score $w_j^{(z)}$ to each component $c_j \in C$. Then, as per Equation 3, 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`) [31]: This method extends the Integrated Gradients method [104]—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 [31] for a formal description.

- **Internal Influence** (`II`) [70]: Similar to `NC`, this method also adapts Integrated Gradients [104] 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 [70] for a formal description.

We implement the first two baselines (`LOO` and `GP`) from scratch[3] and use the `captum` library [65] [4] 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.

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 7). 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 $\{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$ 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×, resulting in a direct speedup while only reducing the predictive power of COAR attributions by a small amount.

3. Our code is available at `https://github.com/MadryLab/modelcomponents`
4. Github repository: `https://github.com/pytorch/captum`

Data loading. We use the `FFCV` library[5] [69] 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 (**??**)—requires solving a linear regression problem (Equation 2) for each example z. We parallelize this step by using the `fast-l1` package[6], 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.

5. Github repository: https://github.com/libffcv/ffcv
6. Github repository: https://github.com/MadryLab/fast_l1

Appendix 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 [92] evaluated on the next-token prediction task and (b) Phi-2 [72] 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. [90], 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 (4) over all tokens in a given sequence.

Model and dataset. In this experiment, we consider the GPT-2 model[7] [92], 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[8] [33], 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 (**??**) 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 8a, 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 (3) using its COAR attributions $\boldsymbol{\theta}^{(\boldsymbol{z})}$, as defined in Equation 5. In Figure 8b, 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 (4) as the model output function for this zero-shot binary classification task.

7. `https://huggingface.co/gpt2`
8. `https://huggingface.co/datasets/roneneldan/TinyStories`

Model and dataset. We consider the Phi-2 model[9] [72] 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[10] [23], 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. [40][11], 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 (**??**) 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 9a plots the correlation between ground-truth component counterfactuals $f_M(z, \cdot)$ and the corresponding COAR estimate (3) of a random BoolQ example z. The histograms in Figure 9b 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.

9. https://huggingface.co/microsoft/phi-2
10. https://huggingface.co/datasets/google/boolq
11. https://github.com/EleutherAI/lm-evaluation-harness

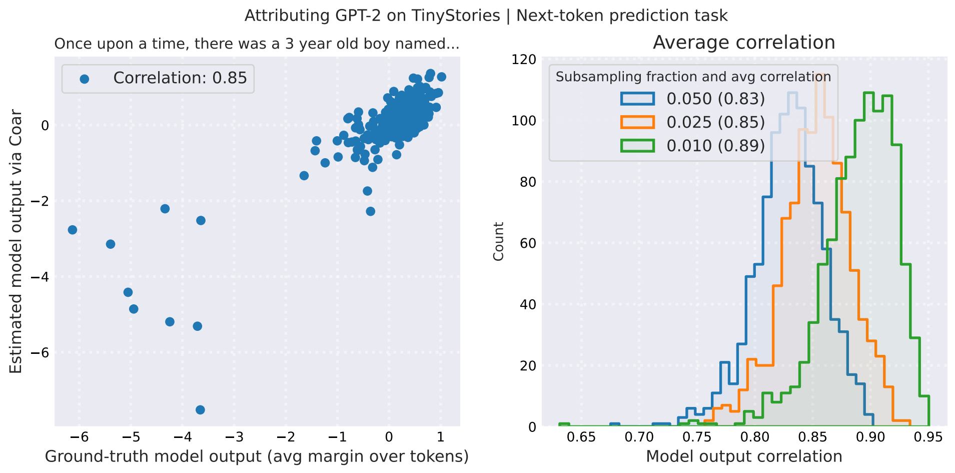


Figure 8: **Evaluating COAR on GPT-2.** We apply COAR to the GPT-2 model [92] on the TinyStories dataset [33]. The resulting component attributions are predictive of component counterfactuals. The left plot shows that component attributions can estimate the effect of ablating components on the average 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.

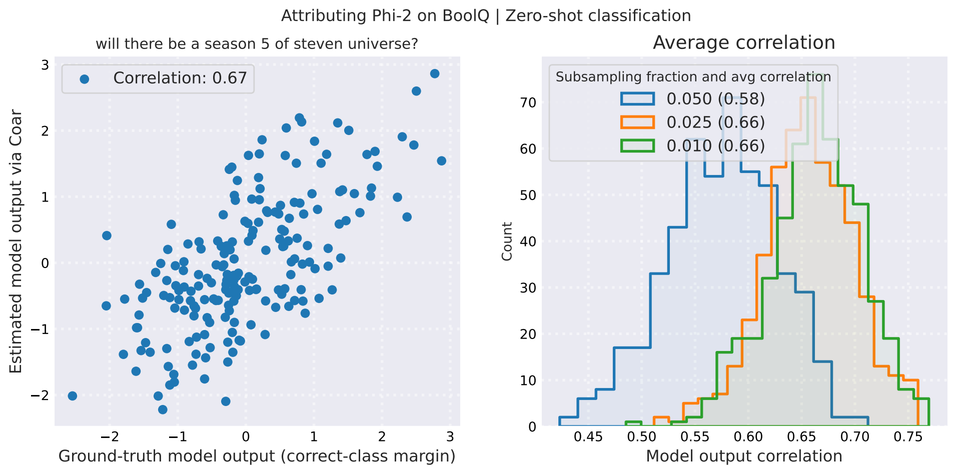


Figure 9: **Evaluating COAR on Phi-2.** We apply COAR to the Phi-2 model [62] on the BoolQ dataset [23]. The resulting component attributions are predictive of component counterfactuals. The left plot shows that component attributions can estimate the effect of ablating components on the average correct-class 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.

Appendix 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 [53]. as shown in Figure 10, 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 11 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 Appendix D.3—see first row of Figure 12. 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 [63], 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 12 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 12 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 [76]. 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 13, 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 14 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 14 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 15 and on ImageNet examples in Figure 16. 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 17, 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 17 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 [34], 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 18 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).

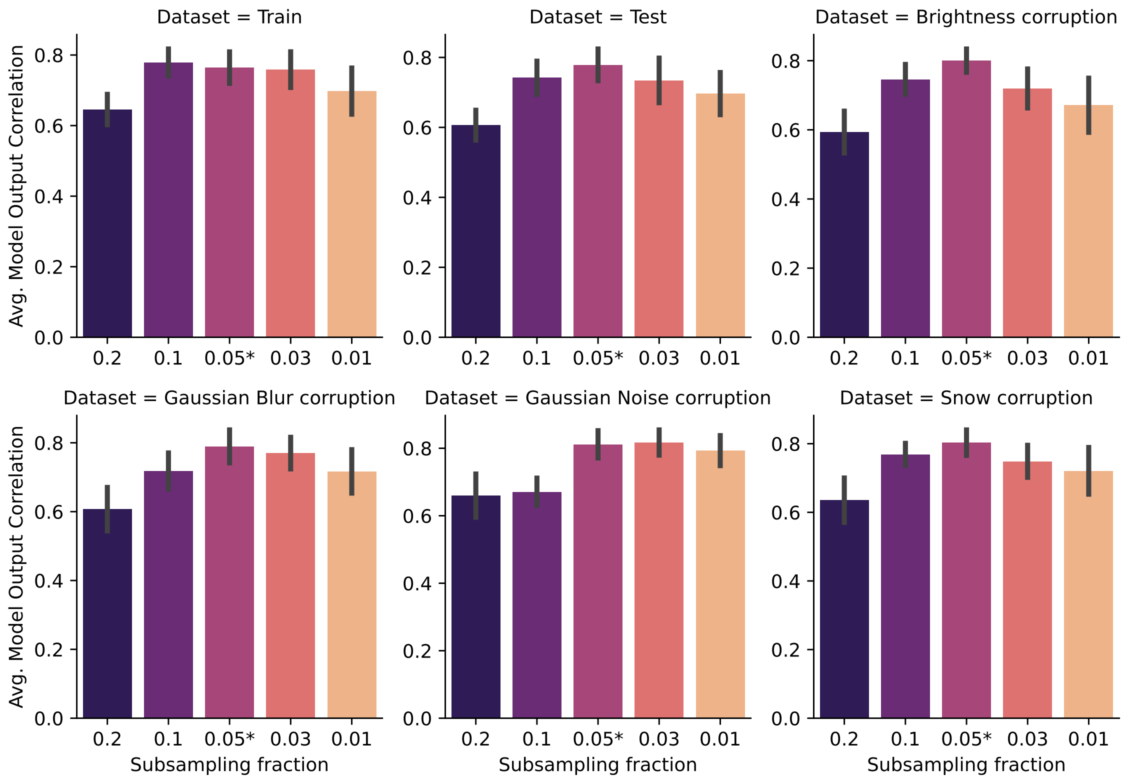


Figure 10: **Do COAR attributions generalize to out-of-distribution examples?** COAR attributions remain predictive on the CIFAR-10 training set and on six corrupted versions of the CIFAR-10 test set [53] over a range of ablation fractions α. See Appendix G.1 for more details.

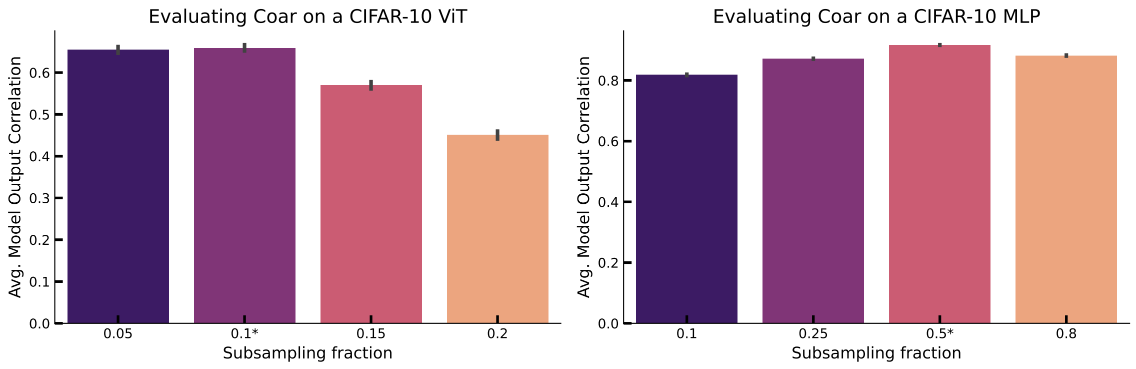


Figure 11: **Do COAR attributions generalize to other model architectures?** COAR attributions yield accurate estimates of component counterfactuals on two additional model architectures: a ViT-based model (left) and a one-layer fully-connected network (right) trained on CIFAR-10. See Appendix G.2 for more details.

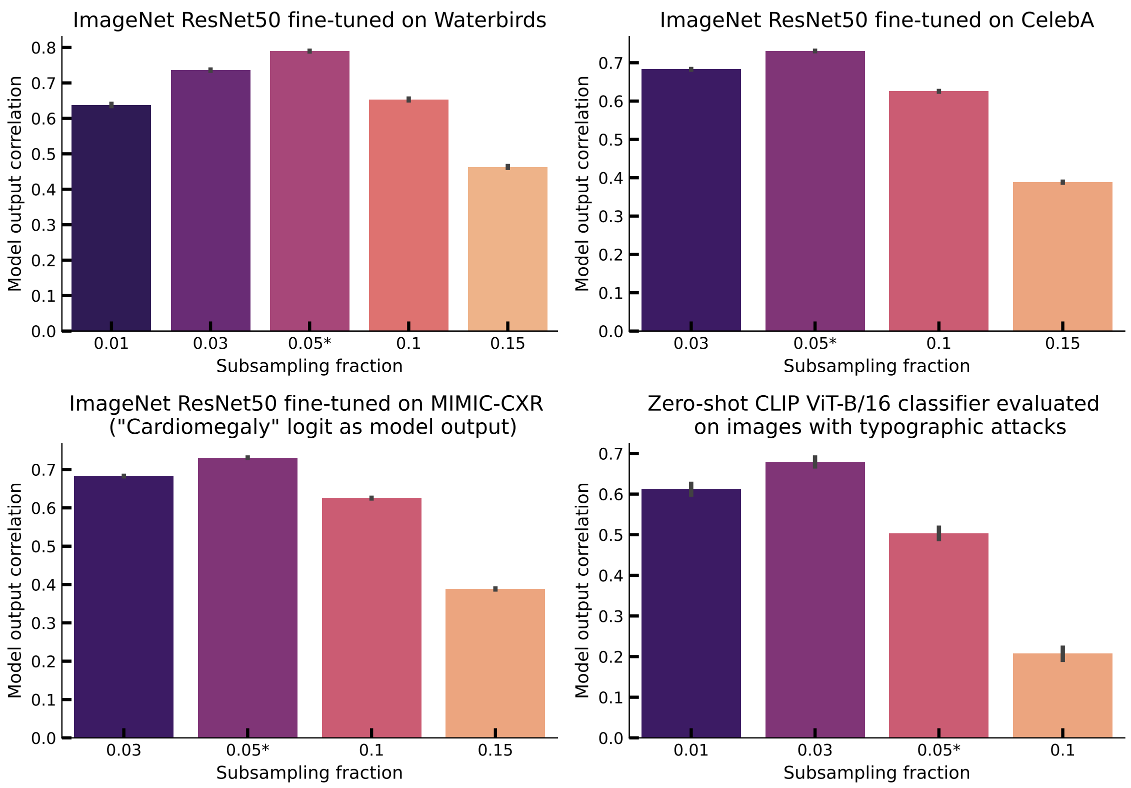


Figure 12: **Evaluating COAR attributions on additional tasks.** We find that component attributions estimated using COAR are predictive on four additional tasks: fine-tuning ImageNet ResNet50 on Waterbirds, CelebA and MIMIC-CXR, and a zero-shot CLIP ViT-B/16 classification task on a dataset containing typographic attacks (Appendix D.4). 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.

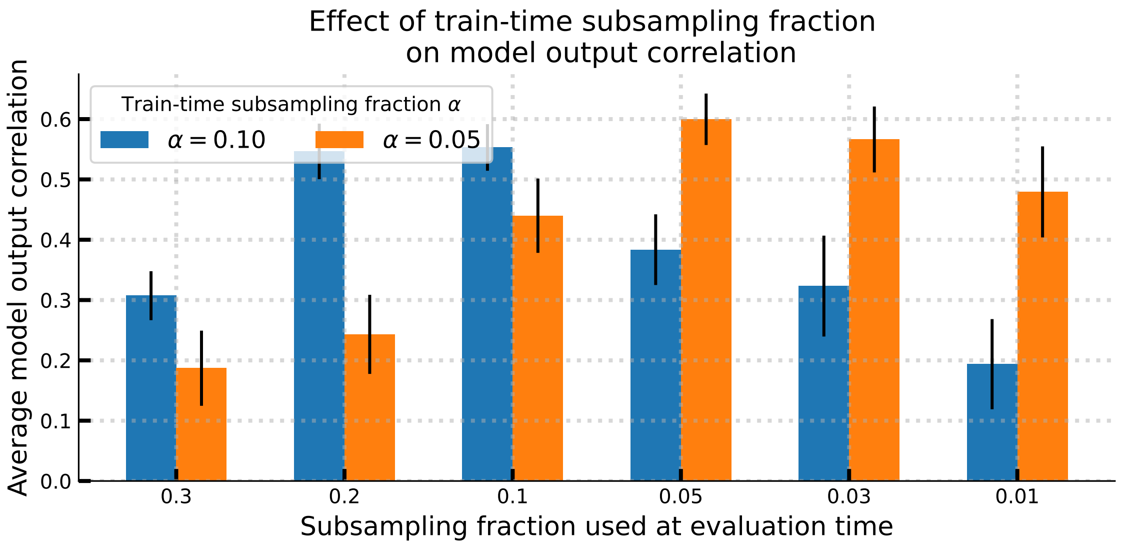


Figure 13: **Comparing COAR attributions estimated with different ablation fractions α.** COAR attributions estimated with different ablation fractions α_{train} attain a different correlation "profile" over α at test time. The correlation between ground-truth component counterfactuals and attribution-based 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.

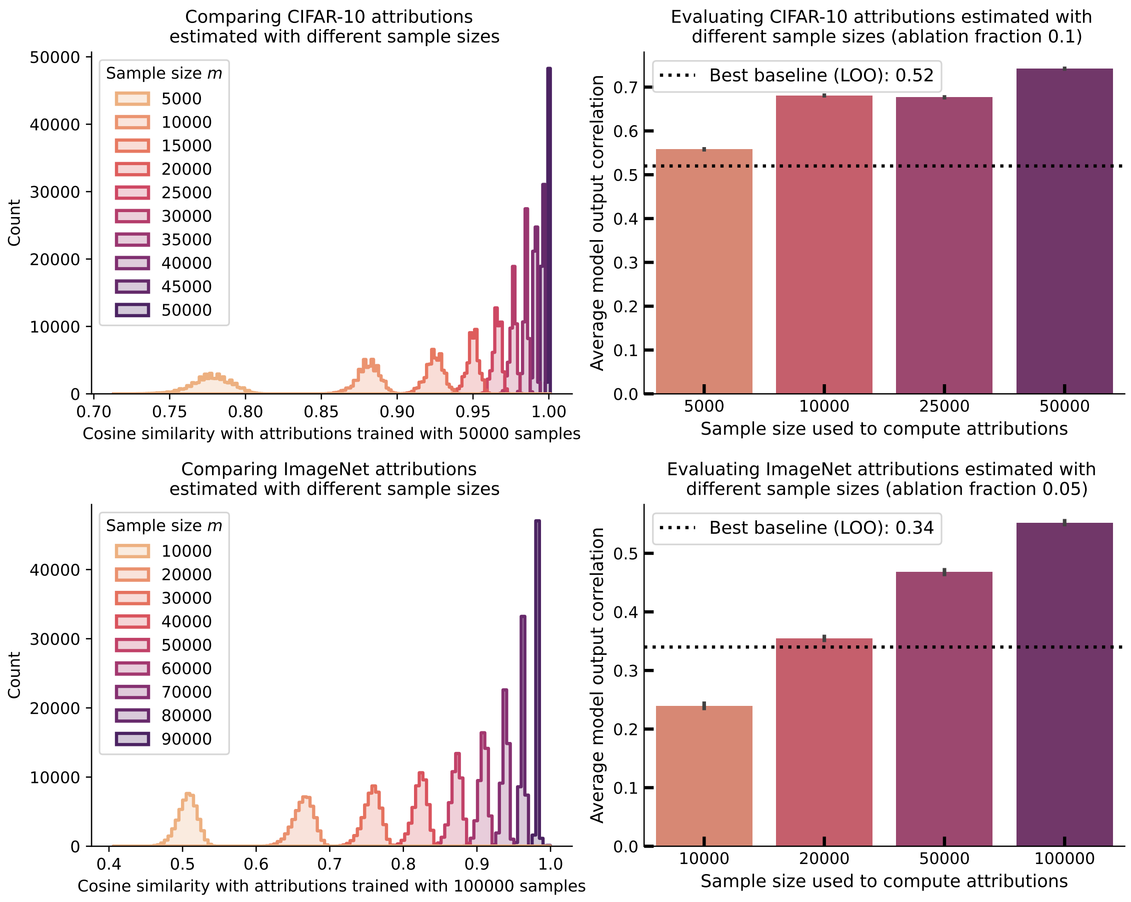


Figure 14: **Comparing COAR attributions estimated with different sample sizes.** COAR attributions for CIFAR-10 ResNet-18 and ImageNet ResNet-50 (Setup A and B respectively in Section 4) estimated with smaller sample sizes m are still predictive of component counterfactuals. On the left, we show that 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).

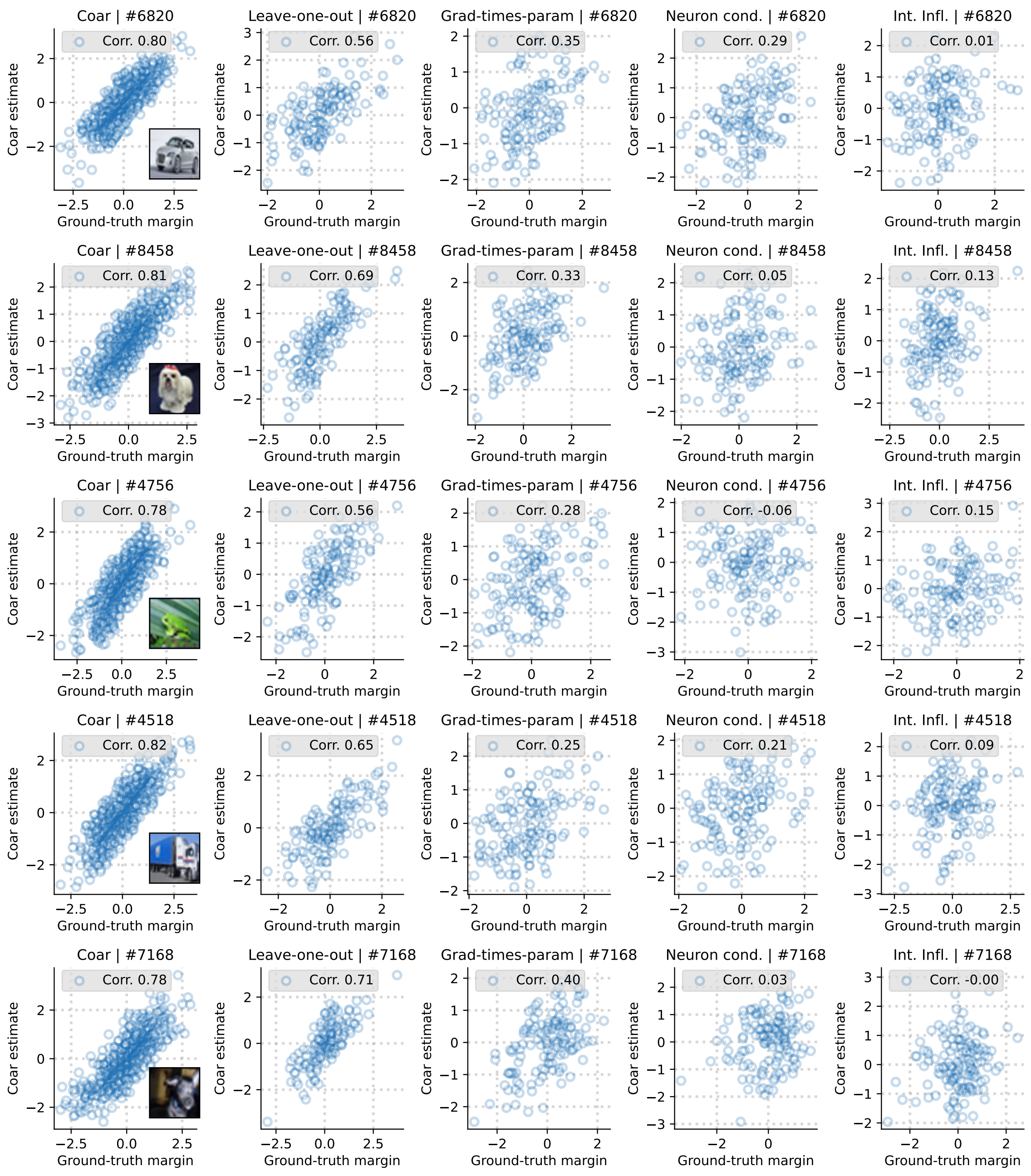


Figure 15: **Additional example-level evaluation of component attributions on CIFAR-10.** Each row corresponds to a different example z randomly picked from the CIFAR-10 test set and each column corresponds to a different attribution method. The left-most subfigure in each row shows that COAR attributions 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.

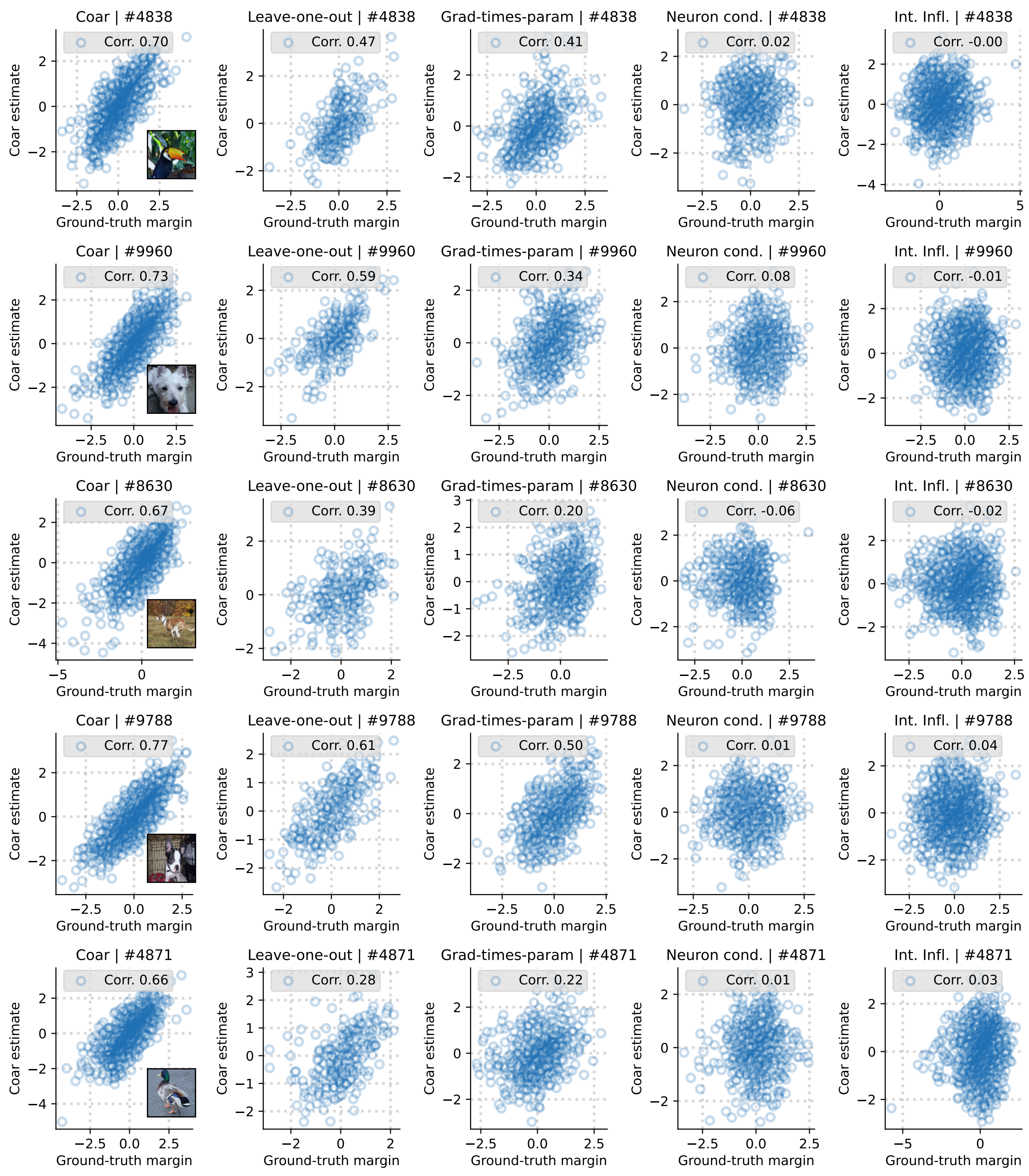


Figure 16: **Additional example-level evaluation of component attributions on ImageNet.** Similar to the results in Figure 15, each row corresponds to a different example z randomly picked from the ImageNet test set. The left-most subfigure in each row shows that COAR attributions and the corresponding 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.

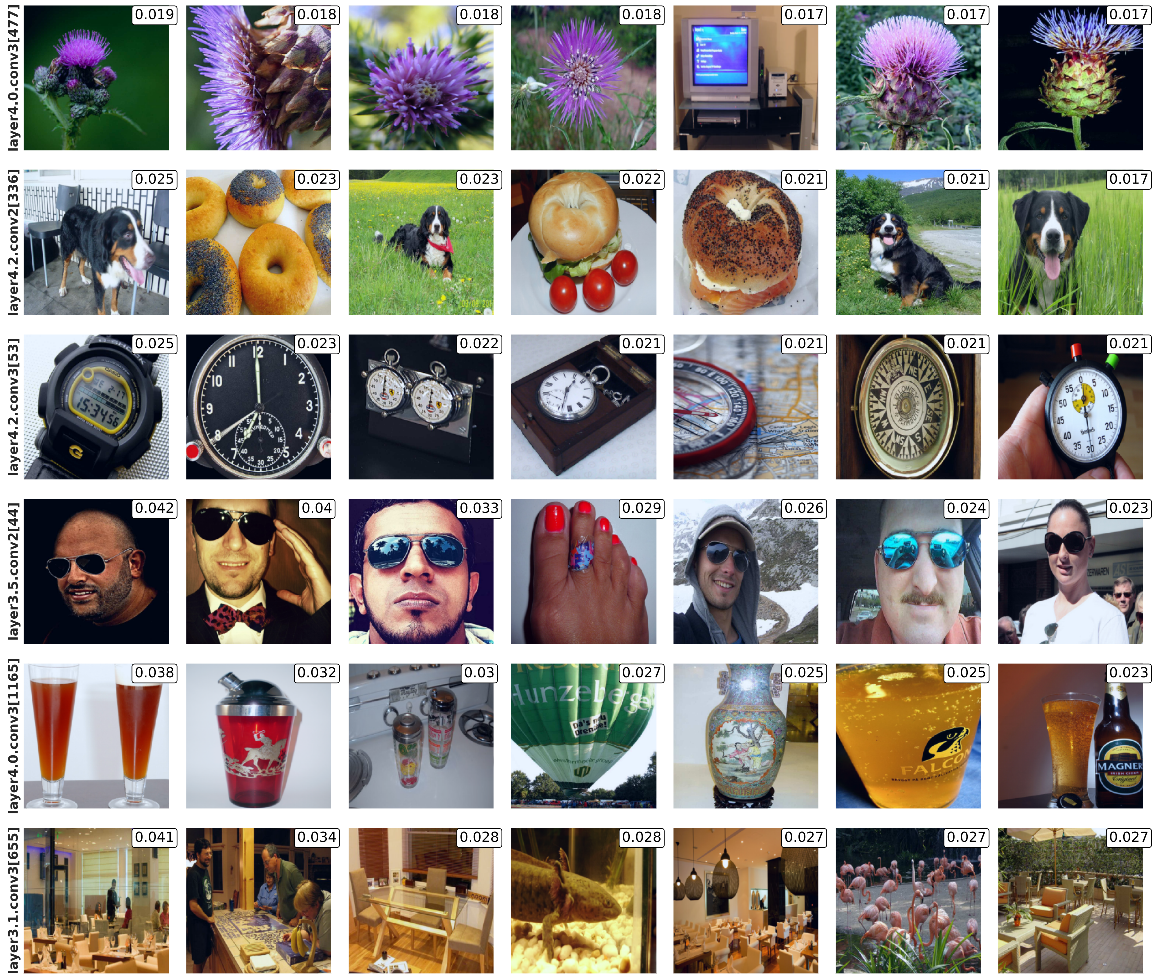


Figure 17: **Visualizing component-specific attributions across examples.** We sample a random set of components from the ImageNet ResNet-50 model (setup B in Section 4) and visualize the examples with the most positive attributions for each component. In general, the examples with the most positive attributions for a given component exhibit visual similarity at different levels of granularity. For example, the first, third and fifth row in Figure 17 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 [34], 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.

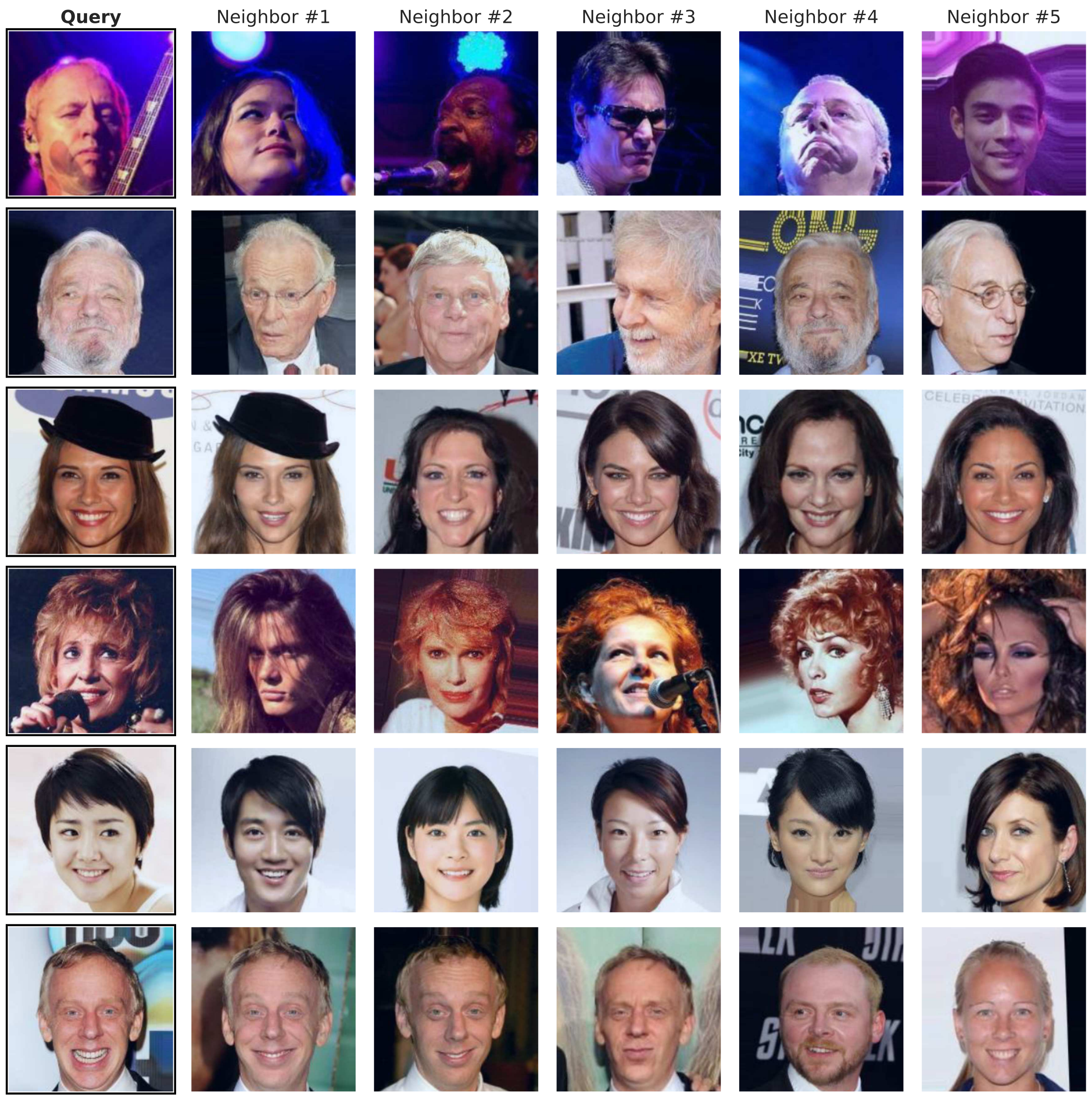


Figure 18: **Visualizing nearest neighbors in COAR attribution space.** We also use component attributions as feature embeddings in order to visualize the five nearest neighbors of examples from the CelebA dataset in "component attribution" space. Intuitively, this technique allows us to identify examples 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).

Appendix H. Additional evaluation of COAR-EDIT

We use COAR-EDIT in five different editing tasks: correcting misclassifications (§D.1); forgetting a class (§D.2); improving subpopulation robustness (§D.3); localizing backdoor attacks (§D.5); and improving robustness to typographic attacks (§D.4). In this section, we provide additional details and/or supplementary experiments for each task.

H.1. Editing individual predictions

Experiment details. In Appendix D.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 (**??**) that increase the correct-class margin (4) 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 20, we reproduce the experiment in Appendix D.1 on additional ImageNet examples misclassified by a ResNet-50 model. In Figure 19, we use COAR-EDIT to similarly fix misclassifications of a ResNet-18 model on the CIFAR-10 dataset. In Figure 21, 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 Appendix D.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 22. 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.2, 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 (**??**) that decrease the average correct-class margin (4) 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 23, we reproduce the COAR-EDIT experiment in Appendix D.2 on three additional ImageNet classes: "folding chair", "military uniform", and "revolver". Like in Figure 3, 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 Appendix D.3, 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. [97]. 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 (4) 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.5, 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 (**??**) that increase the correct-class margin (4) of the target examples without impacting the average margin over the reference examples.

Additional analysis. Recall that our experiment in Appendix D.5 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 24. Specifically, we visualize the ablated components (convolution filters in this case) using the input-times-gradient saliency map method from the Captum library [65]. As shown in Figure 24, these visualizations suggest that the ablated components are sensitive to the blue-squared trigger.

H.5. Improving robustness to typographic attacks.

Experiment details. In Appendix D.4 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 [93] 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 [76]. 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 (**??**) that increase the average correct-class margin (4) 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.

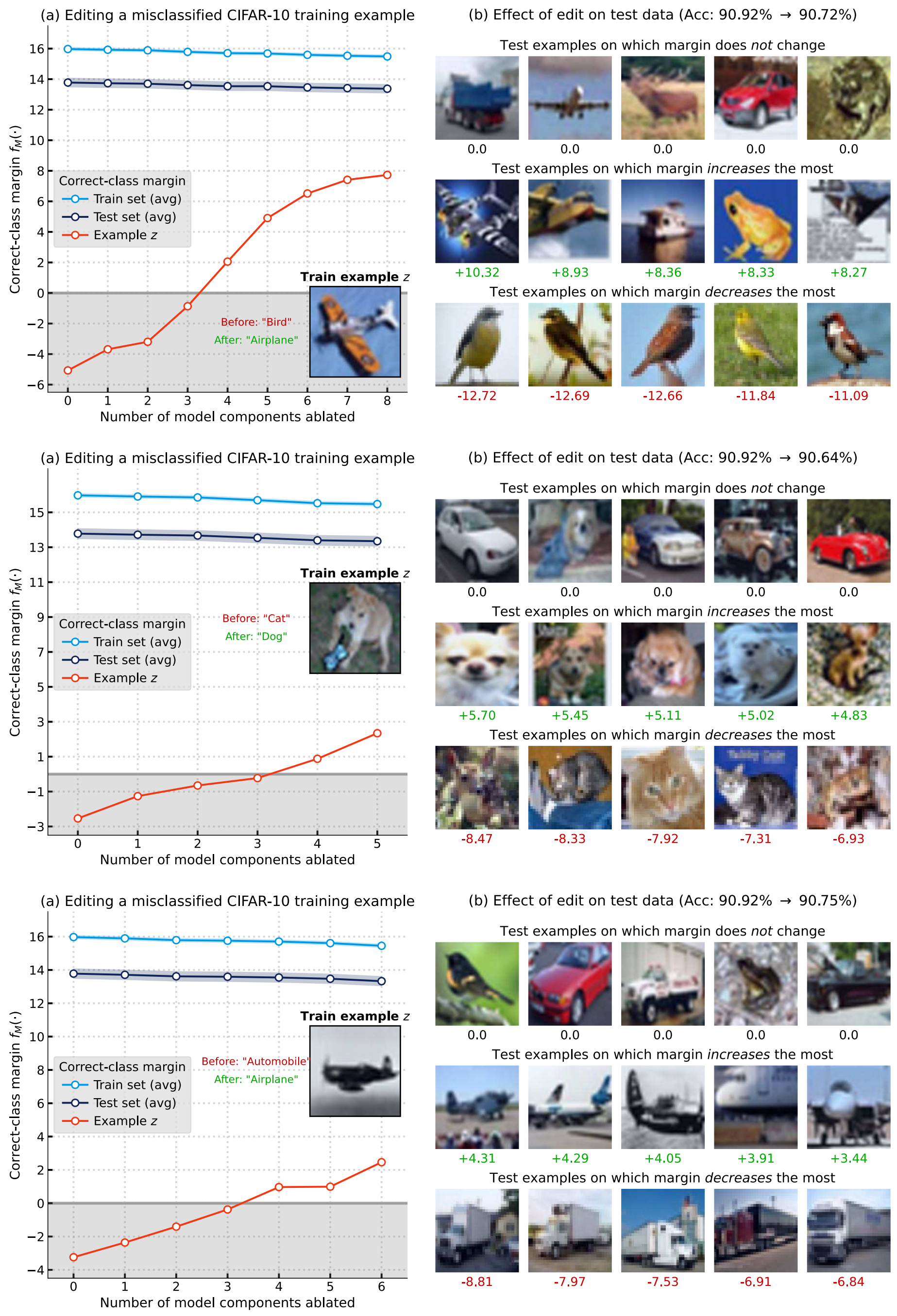
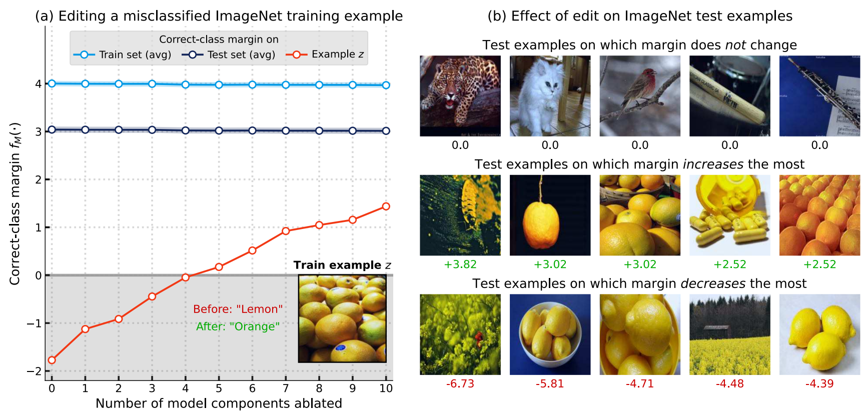
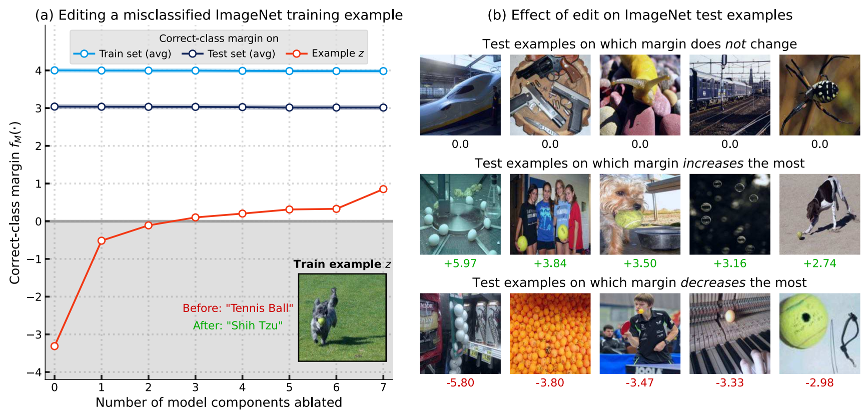


Figure 19: **Correcting misclassified CIFAR-10 examples via COAR-EDIT.** We reproduce the COAR-EDIT experiment from Appendix D.1 on the CIFAR-10 dataset. Specifically, each row corresponds to CIFAR-10 test example that is misclassified by a ResNet-18 model. The left subplot in each row shows how applying COAR-EDIT (by ablating components (**??**)) increases the correct-class margin (4) 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.





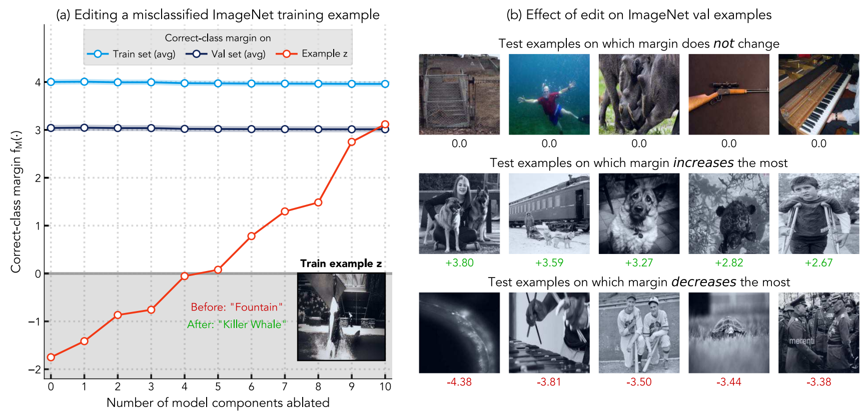
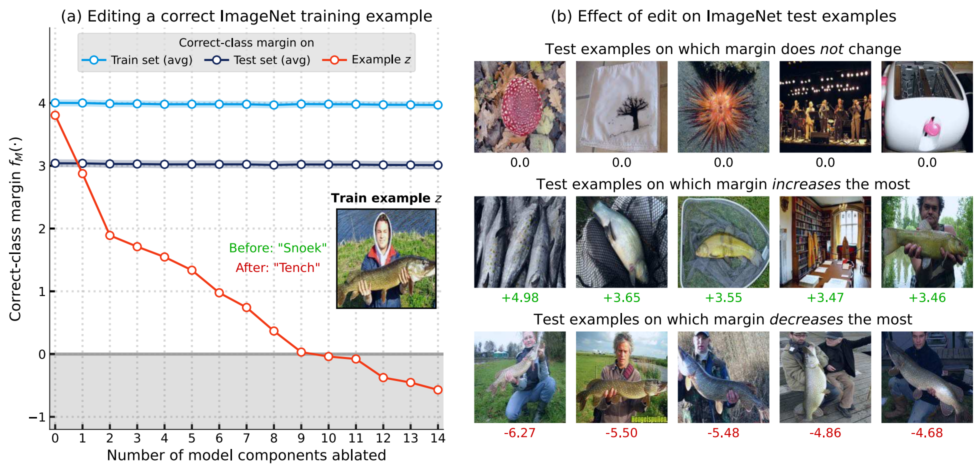


Figure 20: **Correcting misclassified ImageNet examples via COAR-EDIT.** We reproduce the COAR-EDIT experiment from Appendix D.1 on additional ImageNet examples (one per row) misclassified by a ResNet-50 model. The left subplot shows that applying COAR-EDIT (by ablating components (**??**)) increases the correct-class margin (4) 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.



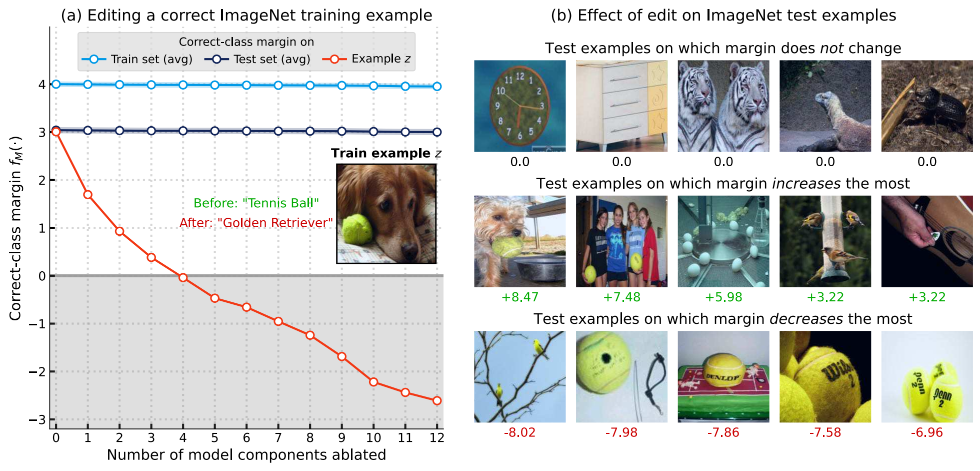


Figure 21: **Adversarially inducing misclassifications on ImageNet examples via COAR-EDIT.** Each row corresponds to an ImageNet test example that is correctly classified by a ResNet-50 model. In the left subplot of each row, we show that applying COAR-EDIT (by ablating the top-k components (**??**)) decreases the correct-class margin (4) 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.

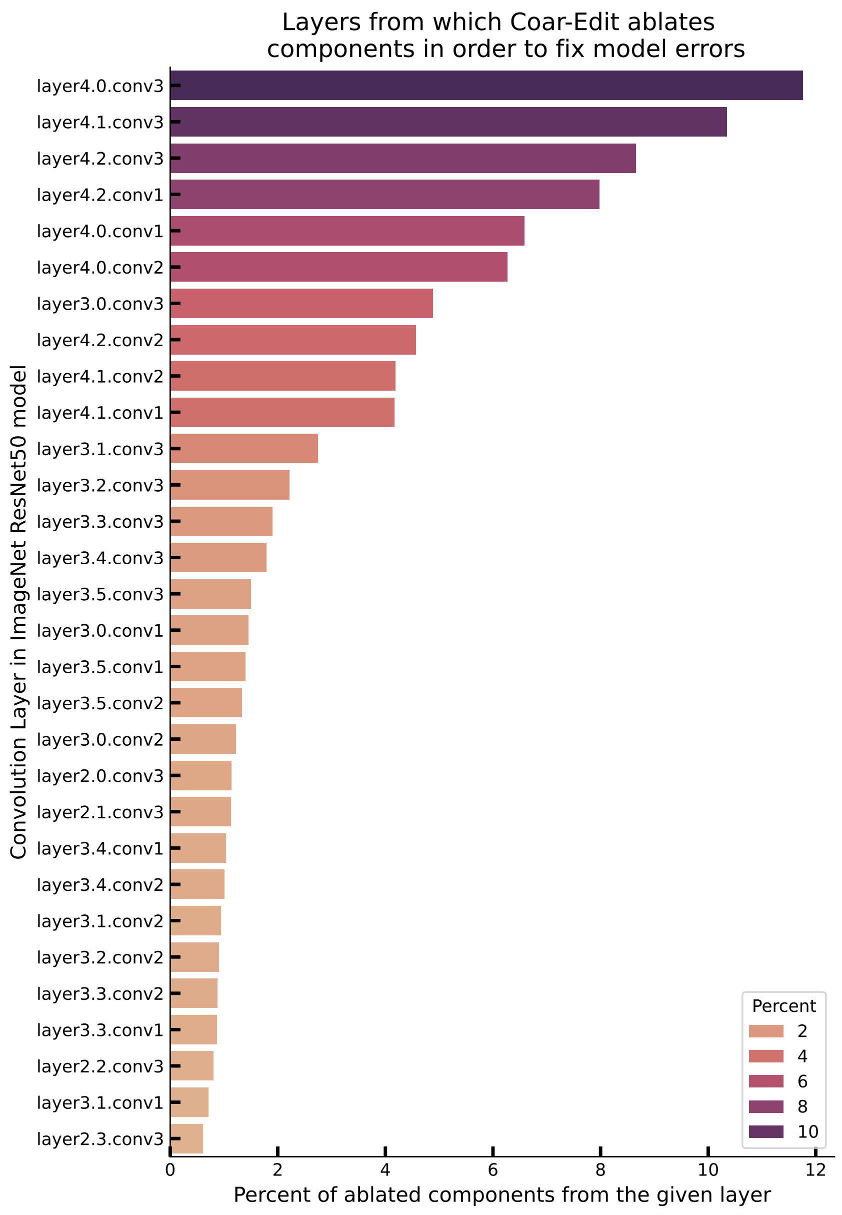


Figure 22: **Which components does COAR-EDIT target to fix model errors?** We analyze the specific convolution layers from which COAR-EDIT ablates components (convolution filters) to correct ImageNet examples misclassified by a ResNet-50 model. On the y-axis, we plot the 30 most common convolution layers 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.

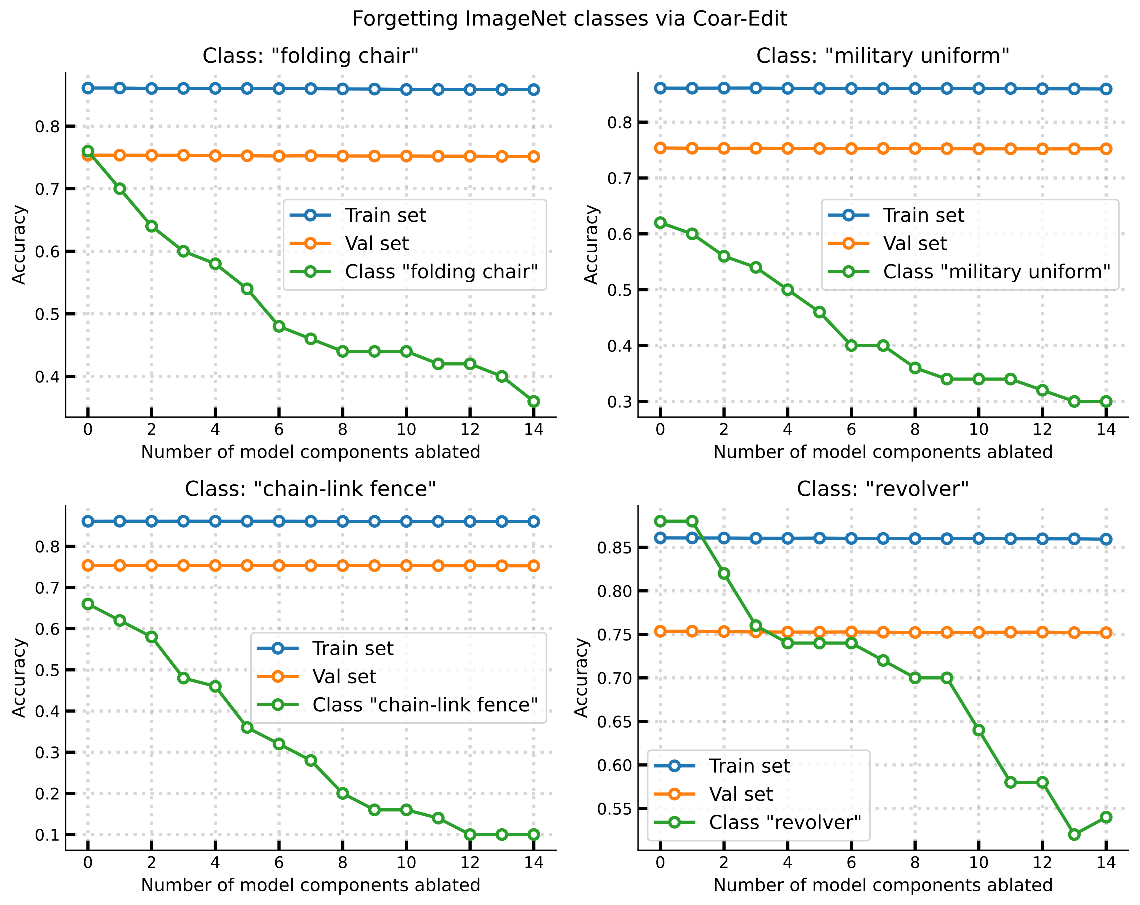


Figure 23: **Forgetting ImageNet classes via COAR-EDIT.** We reproduce the COAR-EDIT experiment from Appendix D.2 on additional ImageNet classes (one per subplot). Specifically, in each subplot, we find that ablating 15 of $22,720$ convolution filters (identified via COAR-EDIT) suffices to significantly 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).

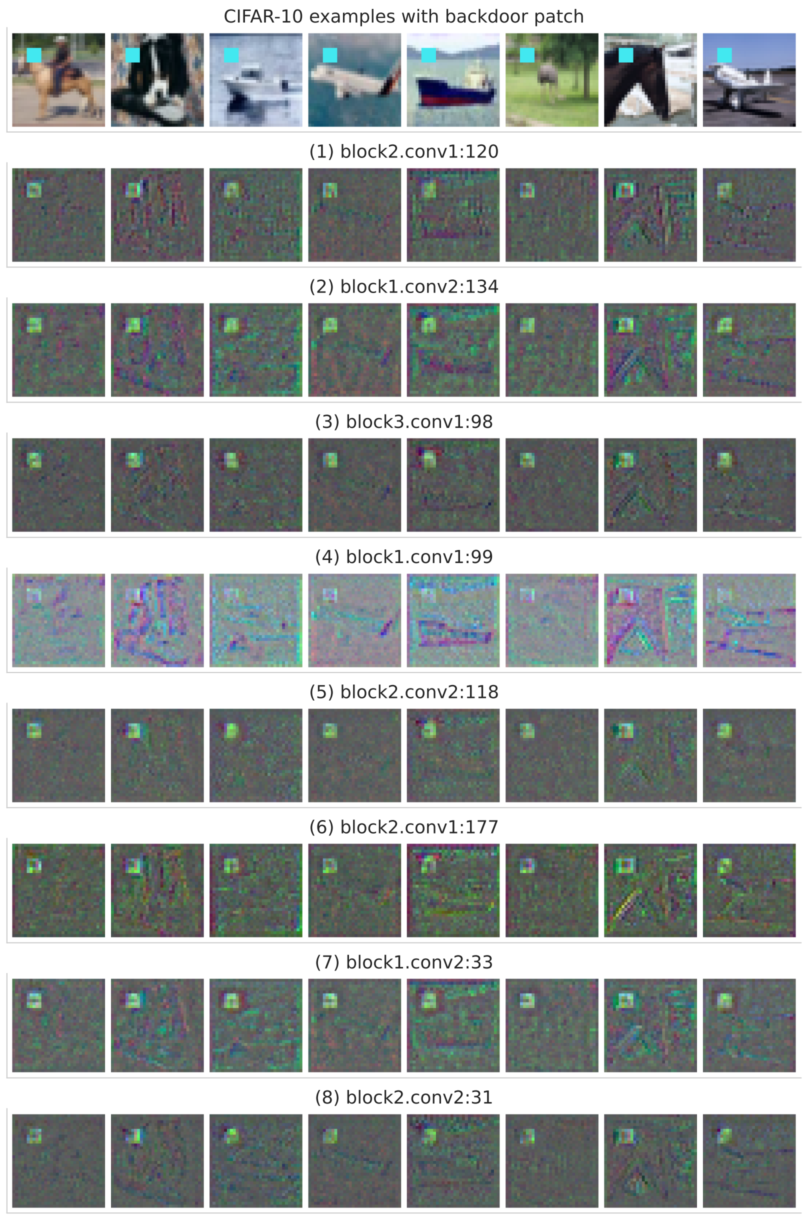


Figure 24: **Visualizing components ablated via COAR-EDIT to mitigate a backdoor attack.** Recall that in Appendix D.5, we used COAR-EDIT to mitigate the effect of a backdoor attack (a blue-squared spurious trigger) on a ResNet-18 model trained on a backdoored CIFAR-10 dataset. Here, we visualize the 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 [65] 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.

Appendix 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 (**??**) 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 (**??**)—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 25 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 25). 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 **??**, 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 **??**; 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 25).
- Then, in Figure 26, 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 27, 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 27a) and that (b) attributions learned with scaling-based ablations are predictive of ground-truth component counterfactuals computed using zero-ablations (Figure 27b). 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 (4) 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 12, we apply COAR to a pre-trained ImageNet ResNet50 model fine-tuned on MIMIC-CXR [63]—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.

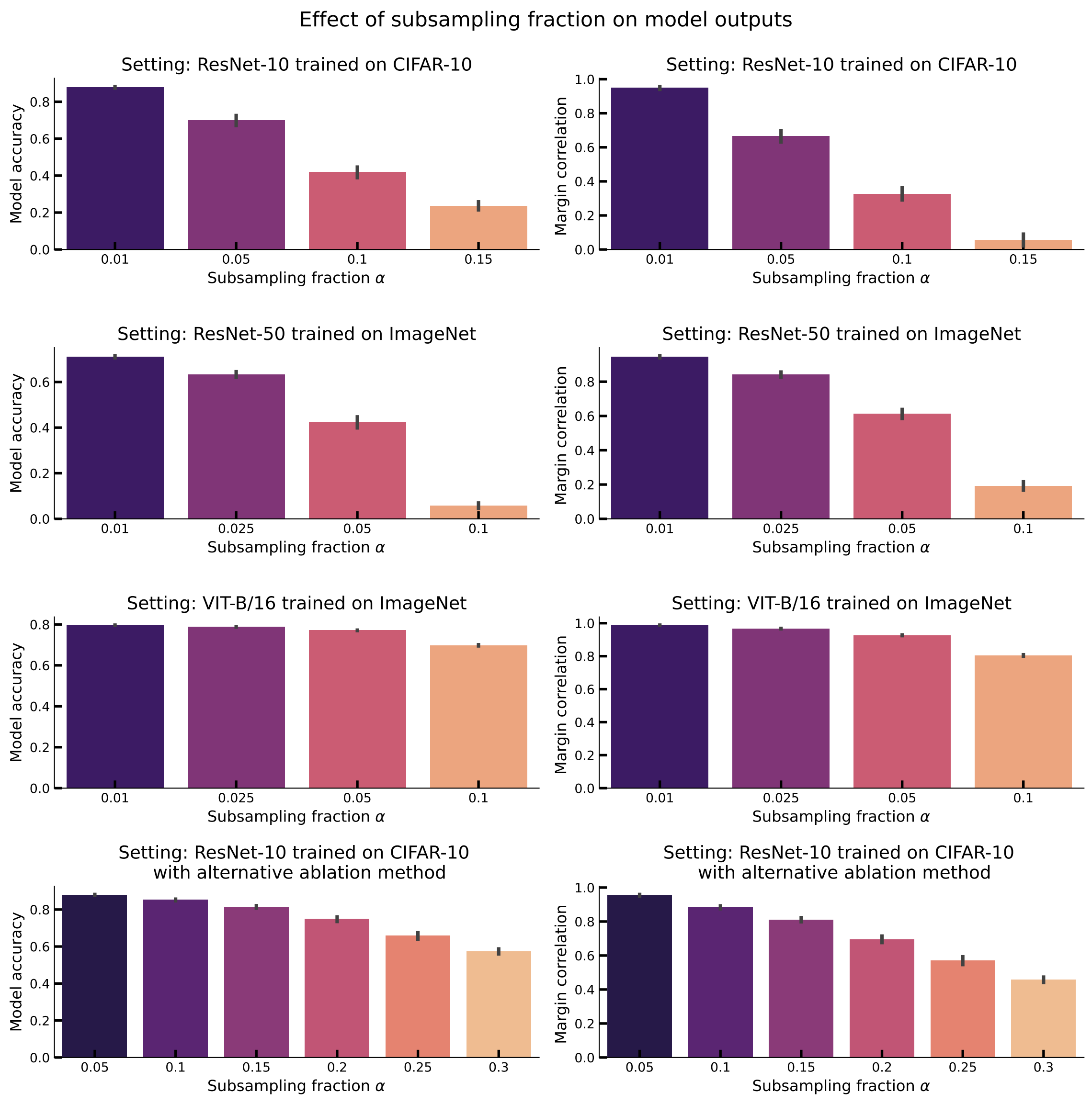


Figure 25: **Effect of ablation fraction α on model outputs.** We evaluate the effect of ablating α-fraction subsets $C' \subset C$ of components (x-axis) on model accuracy (y-axis in the left column) and the correlation between model outputs before and after ablation (y-axis in the right column). In 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.

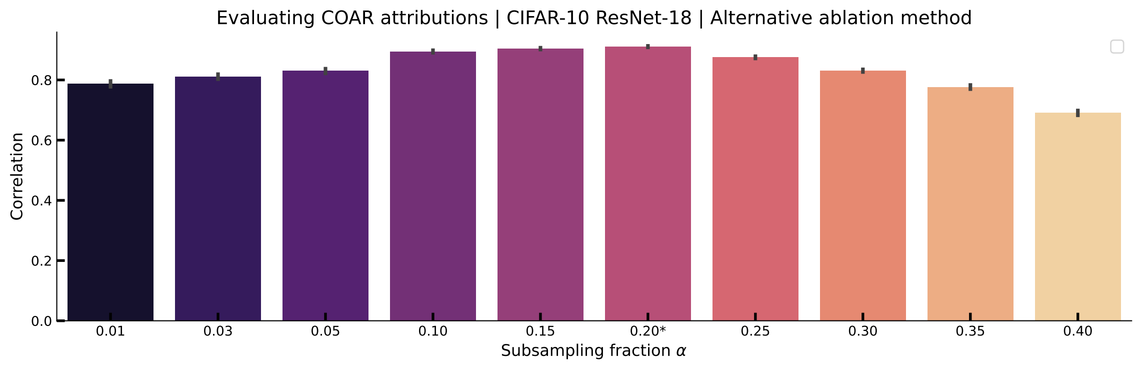


Figure 26: **Effect of ablation method on COAR attributions.** We estimate COAR attributions for a CIFAR-10 ResNet-18 model using an alternative ablation method that scales down the activations of a subset of components $C' \subset C$ by a factor of γ (0.5 in this case) instead of setting them to zero. The resulting attribution-derived estimates (3) exhibit high correlation (y-axis) with ground-truth component counterfactuals. See Appendix I.2 for more details.

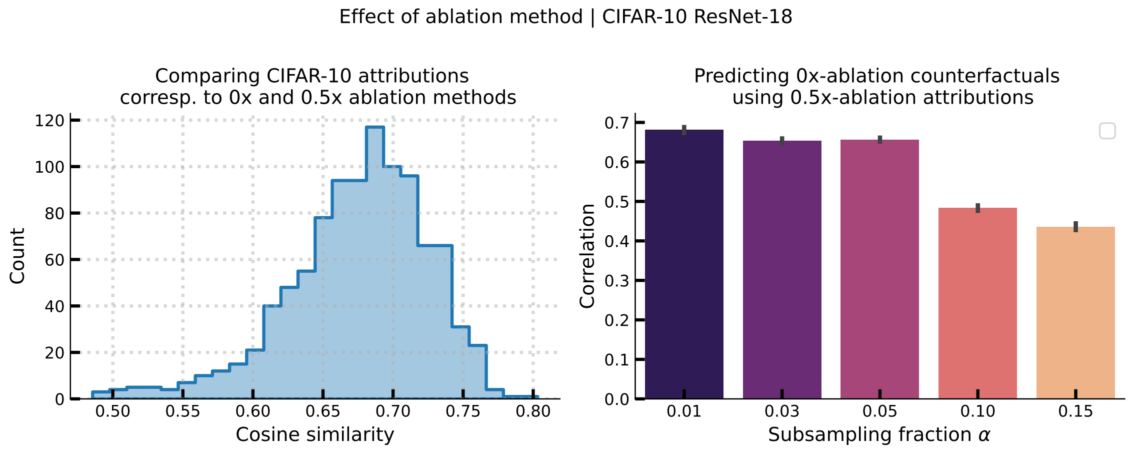


Figure 27: **Comparing COAR attributions estimated with different ablation methods.** We compare COAR attributions on a CIFAR-10 ResNet18 model computed with the zero-ablation method **??** to attributions computed with the alternative ablation method described in Appendix I.2. The left plot shows that the paired attributions (corresponding to each example) exhibit high cosine similarity. The right plot shows that the counterfactual estimates (3) 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.