Editing a classifier by rewriting its prediction rules

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a methodology for modifying the behavior of a classifier by *directly rewriting* its prediction rules. Our method requires virtually no additional data collection and can be applied to a variety of settings, including adapting a model to new environments, and modifying it to ignore spurious features.

1 Introduction

At the core of machine learning is the ability to automatically discover predictive correlations in data. However, there is mounting evidence that not all of the correlations that the resulting models rely on to make their predictions are actually meaningful [65, 7, 60, 1, 68, 10, 23]. In particular, such correlations can arise from biases in the data used to train models, and thus may not reflect the real world—e.g., a pasture, while typical, need not be present in every cow image [7]. Consequently, model designers might want their models to avoid such correlations, as while these may be predictive in some scenarios, they can be irrelevant or misleading in others. This raises the question:

How can we modify the way in which a given model makes its predictions?

The canonical approach for modifying a model post-hoc is to intervene at the data level: collect additional input-label pairs that capture the desired behavior and use them to further train the model. Unfortunately, collecting such data can be challenging: how do we get cows to pose for us in a variety of environments? Furthermore, data collection is ultimately a very indirect way of specifying the intended behavior of a model. Even when the data has been carefully curated to reflect a given real-world task, models still end up learning unintended prediction rules from it [49, 65, 66, 9].

Our Contributions

The goal of our work is to develop a toolkit that enables users to *directly modify* the prediction rules learned by an (image) classifier. Concretely:

Discovering prediction rules. Before intervening on a model, one would first need to gain visibility into how the model makes its decisions—e.g., to discover the failure modes that need to be corrected. To this end, we design an automated pipeline that can help identify high-level concepts (e.g., "wheel") that a model relies on to recognize images of a certain class (e.g., "car"). Specifically, our pipeline leverages state-of-the-art methods for instance segmentation and style transfer to identify concepts which, when transformed, cause the model performance to drop significantly.

Editing prediction rules. We then build on the recent work of Bau et al. [5] to develop a method for modifying a model's prediction rules in a targeted and controlled manner, with essentially *no* additional data collection. We demonstrate the effectiveness of this approach in adapting models to the concept-level transformations that result from our prediction rule-discovery pipeline described above. Fundamentally, our method changes the behavior of the model on occurrences of a particular concept beyond just the specific classes it encounters during the edit. For instance, we can modify

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

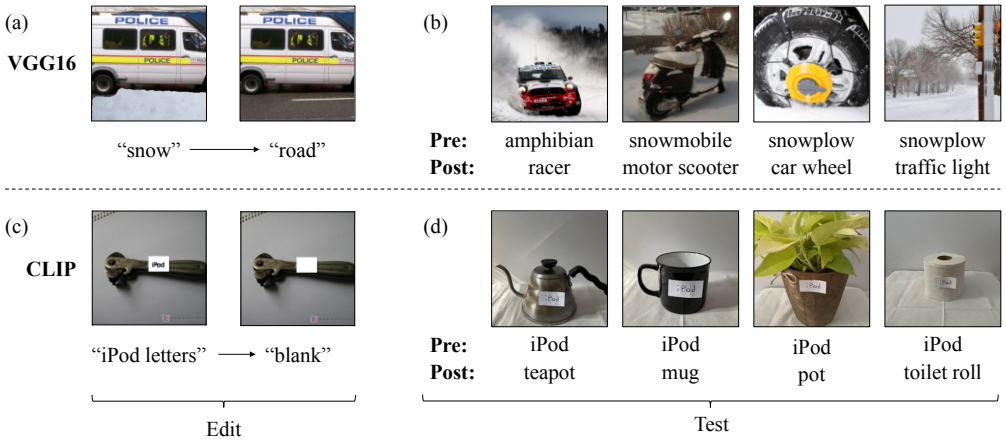


Figure 1: Editing prediction rules in pre-trained classifiers using a *single* exemplar. (a) We edit a VGG-16 ImageNet classifier to map the representation of the concept "snow" to that of "asphalt road". (b) This edit corrects systematic classification errors on snowy scenes for various ImageNet classes. (c) We edit an OpenAI CLIP model such that the text "iPod" maps to a blank area. (d) This change makes the model robust to the typographic attacks from Goh et al. [26].

a pre-trained ImageNet classifier to correctly recognize vehicles with unusual "wooden wheels" by teaching it to treat a "wooden wheel" in a "car" image as it would a regular wheel—see Figure 4 for an illustration. In contrast, the typical approach of modifying models via fine-tuning consistently fails to generalize in the same way (Section 4).

Real-world demonstrations. Finally, we apply our approach to two scenarios reflecting concept-level transformations that could arise in the real world (Section 5). First, we focus on adapting an ImageNet classifier to a new environment, namely, recognizing actual photographs of vehicles on snowy roads. Second, we consider the recent "typographic attack" of Goh et al. [26] on a zero-shot CLIP [50] classifier: attaching a piece of paper with "iPod" written on it to various household items causes them to be incorrectly classified as "iPod." We find that our approach significantly improves model performance in both settings, using only a *single synthetic* example—cf. Figure 1.

Overall, our findings demonstrate that there is a rich design space around *directly* rewriting a model's prediction rules and that this approach can lead to a powerful model debugging toolkit. To facilitate future research in this direction, we will release the code for the tools we developed.

2 Discovering Prediction Rules

Before we can change how our models behave, we must first get a better sense of how they make predictions. However, completely characterizing the prediction mechanisms of a complex model can be challenging; this is an active focus of current research in the field of interpretability (discussed further in Section 6). Instead, our goal here will be to obtain a coarse understanding of model behavior by approximating it with simple, concept-level prediction rules. For instance, we might want to know whether a classifier depends on the presence of a "wheel" to recognize a "car".

Our pipeline for discovering such prediction rules revolves around input *counterfactuals*—a primitive commonly used in causal inference [48] and interpretability [28, 29, 6]. Counterfactuals can be used to identify the data features that the model uses to make its prediction on a given input: by assessing how its prediction changes when the input is modified along a particular axis. In our case, we use counterfactuals to glean how the model's prediction depends on high-level concepts in the image. For example, to understand whether the model relies on the "wheel" to recognize a "car" in an image, we will evaluate it on the same image with a different (transformed) "wheel". We now discuss our approach to generate such counterfactuals (Section 2.1) and then describe how they can be used to gain insight into a model's prediction rules (Section 2.2). Our complete rule-discovery pipeline is illustrated in Figure 2 (see Appendix A.4 for additional details).

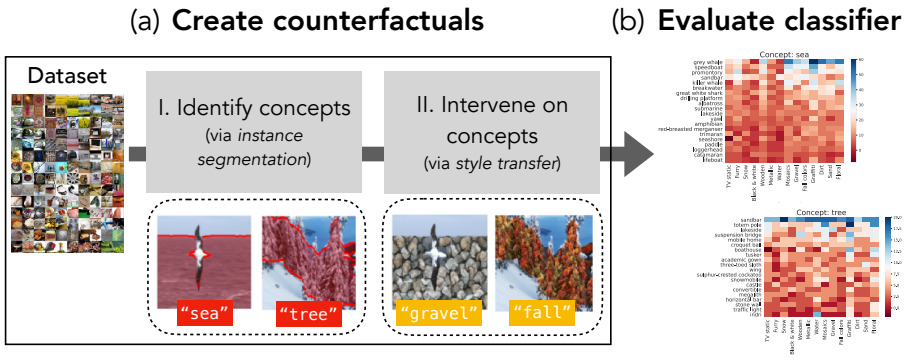


Figure 2: Prediction-rule discovery pipeline. To pinpoint the high-level concepts that have an impact on model predictions, we (a) synthesize counterfactual images in which a given concept in the image (detected via instance segmentation) is modified (via style transfer). Then (b) the impact of this concept-level transformation on classifier predictions is measured. For example, we see that the model relies on "sea" to predict well on "albatros", and on "tree" to detect "sandbar".

2.1 Synthesizing concept-level counterfactuals

We generate counterfactuals with respect to a given high-level concept (e.g., "wheel") in two steps:

1. **Concept identification:** First, we find images in which this concept is present, and pinpoint the regions of each image it appears in. In principle, this could be accomplished by manually segmenting images in a fine-grained manner; however this would be quite costly. So, instead, we leverage pre-trained instance segmentation models (e.g., trained on MS-COCO [41] and LVIS [30]) to automatically obtain approximate concept segmentations.
2. **Concept transformation:** Once we have identified where in a given image the relevant concept is present, we need to transform it. We do so by leveraging existing methods for style transfer [22, 24] so as to preserve fine-grained image features and realism. For our analysis, we manually collect a set of realistic textures (e.g., "snow" and "graffiti") which we then use to transform image regions where the concept is present.

These two steps, when combined, allow us to automatically synthesize counterfactuals with realistic *concept-level transformations*, such as "snowy road" or "wooden wheels". Note that this process does not require *any* annotation effort and can be directly applied to new datasets.

2.2 Probing model behavior via counterfactuals

We now evaluate classifiers on counterfactuals created with respect to various concept-style pairs, and measure the change in their performance (relative to unmodified images) on a per-class level. This allows us to pinpoint:

The effect of specific concepts. We can measure and compare the influence of a given high-level concept on model performance for various classes—in terms of the accuracy drop caused by the transformed concept. For instance, in Figure 3a, we find that the accuracy of a VGG16 ImageNet classifier drops by 25% on images of "croquet ball" when "grass" is transformed, whereas its accuracy on "collie" does not change. In line with previous studies [71, 52, 53, 3, 68], we also find that background concepts, such as "grass", "sea" and "sand", have a large effect model performance. We can contrast this measure of influence across concepts for a single model (Appendix Figure 9), and across architectures for a single concept (Appendix B.1). Finally, we can also examine the effect of the style used to transform a concept—cf. Figure 3b.

Per-class prediction rules. If we focus on the model's predictions for counterfactual inputs belonging to a single class, we can identify high-level concepts that it relies on for performing well on said class. It turns out that aside from the main image object, ImageNet classifiers also heavily depend on commonly co-occurring objects [62, 66, 9] in the image—e.g., the objects "dress" for the class "groom", "person" for the class "tench" (*sic*), and "road" for the class "race car" (cf. Figure 3c and Appendix Figure 11). We can also examine which concept-level transformations hurt model perfor-

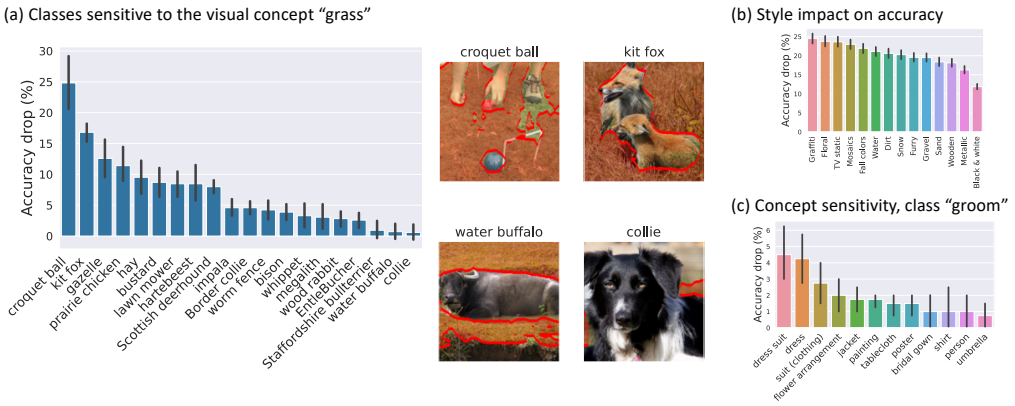


Figure 3: Model sensitivities diagnosed using our pipeline in a VGG16 classifier trained on ImageNet. (a) The accuracy drop induced by transformations of the concept "grass" highlights classes for which the model relies on this concept: e.g., a "croquet ball" is not accurately recognized if "grass" is not present, while "collie"s are not affected. (The twenty classes for which the visual concept is most often present are shown.) (b) Applying different styles to visual concepts reduces accuracy by varying amounts. (c) Visual concepts that cause accuracy losses for a given class can highlight context-dependent rules: e.g., the class "groom" is sensitive to the presence of "dress."

mance the most—e.g., we find that making "plants" "floral" hurts accuracy on the class "damselfly" 15% more than making the "plants" "snowy".

3 A Toolkit for Editing Prediction Rules

As we saw in Section 2, models often pick up context-specific correlations in the data—e.g., using the presence of "road" or a "wheel" to predict "car". Such prediction rules could hinder models when they encounter novel environments (e.g., snow-covered roads), and confusing or adversarial test conditions (e.g., cars with wooden wheels). As a result, after identifying undesirable prediction rules, a user might want to modify them before deploying their model in the wild.

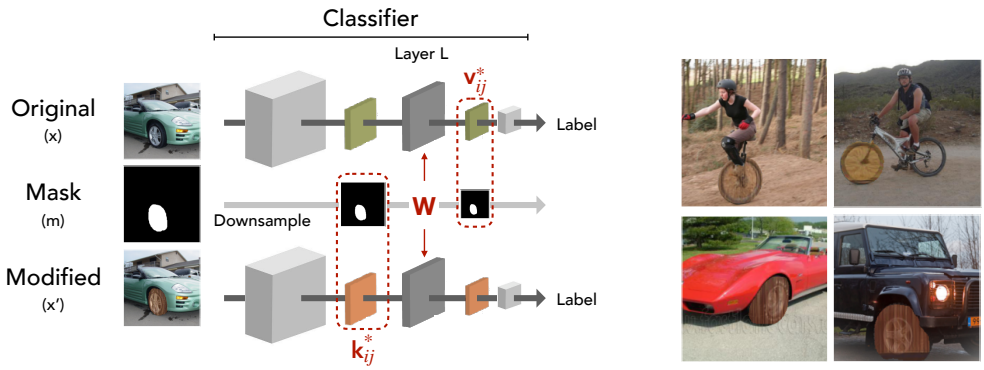
The canonical approach to modify a classifier post hoc is to collect additional data that captures the desired deployment scenario, and use it to retrain the model. However, even setting aside the challenges of collecting such data, it is not obvious a priori exactly what effect such adaptation (e.g., via fine-tuning) will have on the model's prediction rules. For instance, if we fine-tune our model on "cars" with wooden wheels, will it now recognize "scooters" or "trucks" with such wheels?

The goal of this work is to instead develop a more direct way to modify a model's behavior by rewriting its prediction rules in a targeted manner. Ideally, in our previous example, we would like to enable the classifier to recognize vehicles with wooden wheels by simply teaching it to treat *any* wooden wheel as it would a standard one. Before describing our approach in Section 3.2, we first provide a brief overview of recent work by Bau et al. [5] which forms the basis for our approach.

3.1 Background: Rewriting generative models

Bau et al. [5] developed an approach for rewriting a deep generative model, enabling a user to replace all occurrences of one selected object (say, "dome") in the generated images with another (say, "tree"), without changing the model's behavior in other contexts. The approach is built on the observation that, using a handful of example images, we can identify vectors in the model's representation space that encode a specific high-level concept [37, 5]. Leveraging this, Bau et al. [5] treat each layer of the model as an *associative memory*, which maps the concept vector at each spatial location in its input (which we will refer to as the *key*) to a concept vector in its output (which we will call the *value*). In the simplest case, a linear layer with weights $W \in \mathbb{R}^{mxn}$ transforms the key $k \in \mathbb{R}^n$ to the value $v \in \mathbb{R}^m$.

Observe that in this setting, one could perform a rewrite by modifying the layer weights from W to W' so that $v^* = W'k^*$, where k^* corresponds to the old concept that we want to replace, and



(a) Edit (b) Test

Figure 4: Overview of our pipeline for directly editing the prediction-rules of a classifier. The edit in (a) seeks to modify the network to perceive wooden wheels as standard ones, using a small set of exemplar images (say from class "car"). To achieve this, we to first obtain the keys k^*_{ij} corresponding to the new concept (here, "wooden wheel"), and the values v^*_{ij} corresponding to the original concept (here, "standard wheel") in the input and output representation space of a layer L respectively. We then update the weights W of the layer to enforce this new key-value association (1). (b) To test our editing technique, we measure the improvement in model performance on test instances (from any class) containing the new concept—in this case, example images of vehicles with "wooden wheels".

v^* the new concept. For instance, if we wanted to replace "domes" with "trees" in the generated images, we would modify the layer so that the key k^* for "dome" maps to the value v^* for "tree". Consequently, when this value is fed into the downstream layers of the network it would result in a *tree* in the final image. Crucially, this update should change the model's behavior to *every* instance of the concept encoded in k^*—i.e., all "domes" in the images should now be "trees".

To extend this approach to typical deep generative models, two challenges remain: (1) handling non-linear layers, and (2) ensuring that the edit doesn't significantly hurt model behavior in other scenarios. With these considerations in mind, Bau et al. [5] propose making the following rank-one updates to the parameters W of an arbitrary non-linear layer f:

$$\min_{\Lambda} \quad \sum_{(i,j)\in S} \left\| v^*_{ij} - f(k^*_{ij}; W') \right\| \tag{1}$$

$$\text{s.t.} \quad W' = W + \Lambda (C^{-1} d)^\top . \tag{2}$$

Here, S denotes the set of spatial locations in representation space corresponding to the concept of interest, d is the top eigenvector of the keys k^*_{ij} corresponding to locations $(i,j) \in S$ and $C = \sum_d k_d {k_d}^\top$ captures the second-order statistics for other keys k_d. In general, the keys and values in (1) can be obtained not just from various spatial locations in the representations of a single image, but over multiple images containing the concept as well. Intuitively, the goal of this update is to minimally modify the layer parameters to rewrite the desired key-value mapping. We refer the reader to Appendix A.5.1 and Bau et al. [5] for additional details.

3.2 Editing classifiers

We now shift our attention to the focus of this work: editing classifiers. To describe our approach, we will use as a running example the task of enabling classifiers to recognize vehicles with "wooden wheels". Specifically, let us start with a single image x from the dataset, say, from class "car", that contains the concept "wheel". Moreover, let the location of the "wheel" in the image be denoted by a binary mask m.[1] We first create a transformed image x' of a "car" with a "wooden wheel"—i.e., by manually replacing the wheel, or by applying the counterfactual-generation procedure described in Section 2. Next, we would like to apply the approach described above to modify a chosen (potentially non-linear) layer L of the network to rewrite a suitable key-value association. But, we need to first determine what the relevant keys and values are.

[1] Such a mask can either be obtained manually or automatically via instance segmentation (cf. Section 2).

Intuitively, we want the classifier to perceive a "wooden wheel" in the image as it would a standard one. To achieve this, we must map the keys for wooden wheels to the value corresponding to their standard counterparts. Thus, the keys that we want to rewrite correspond to the network's representation of the concept in the *transformed* image directly *before* layer L. Similarly, the values that we want to map these keys to correspond to the network's representation of the concept in the *original* images directly *after* layer L. (The relevant spatial regions in the representation space are simply determined by downsampling the mask to the appropriate dimensions.) Finally, the actual edit is performed by feeding the resulting key-value pairs into the optimization problem (2) to determine the updated layer weights W'—see Figure 4 for an illustration of the overall process.

4 Does Editing Generalize?

We now demonstrate how the methodology described above can be applied to edit vision classifiers—specifically, VGG [61] and ResNet [31] models trained on the ImageNet [13, 54] and Places-365 [73] datasets (cf. Appendix A.2). Our focus will be on improving model generalization to the concept-level transformations from Section 2. Recall that our analysis in that section pinpointed a number of concepts in the data that when transformed—even in fairly natural ways—caused the model performance to drop significantly. We will now use our editing approach to correct these sensitivities.[2]

4.1 Evaluation setup

To edit the model with respect to a particular concept-style pair (say "wheel"-"wooden"), our training set comprises N *exemplars*, i.e., pairs of original and transformed images, (x, x') (cf. Section 3.2) that belong to a single (randomly-chosen) target class in the dataset (e.g., "car"). All other transformed images containing the concept, including those belonging to classes other than the target one, are used for validation and testing (30-70 split). We create two variants of the test set: one using the same style image as the exemplars (i.e., same wooden texture) for the transformation; and another using held-out style images (i.e., other wooden textures). Our baseline is the canonical fine-tuning approach, i.e., directly minimizing the cross-entropy loss on the new data (in this case the transformed images) with respect to the target label. We consider two variants of fine-tuning (cf. Appendix A.5.2): (i) *local* fine-tuning, where we only train the weights of a single layer L (similar to our editing approach); and (ii) *global* fine-tuning, where we also train all other layers between L and the output of the model. It is also worth noting that unlike fine-tuning, our editing approach does not utilize class labels in any way. In each case, we select the best hyperparameters—including the choice of the layer to modify—based on the validation set performance (cf. Appendix A.6.2).

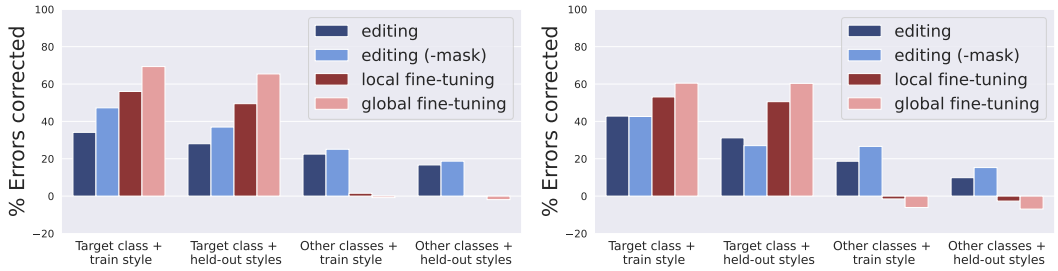
Evaluation criteria. To evaluate the effect of the modification, we need to measure the change in model performance on the transformed examples (e.g., vehicles with "wooden wheel" in Figure 4b). We will only focus on the subset of examples D that were correctly classified before the transformation, since we cannot expect to correct mistakes that do not stem from the transformation itself. Concretely, we will measure the change in the number of misclassifications made by the model on the transformed examples:

$$\frac{N_{pre}(D) - N_{post}(D)}{N_{pre}(D)} \tag{3}$$

where $N_{pre/post}(D)$ denotes the number of transformed examples misclassified by the model before and after the modification, respectively. Note that this metric can range from 100% when rewriting leads to perfect classification on the transformed examples, to even a negative value when the rewriting process causes more mistakes that it fixes.

To quantify the effect of the modification on overall model behavior, we also measure the change in its (standard) test set performance. Since we are interested in rewrites that do not significantly hurt the overall model performance, we only consider hyperparameters that do not cause a large accuracy drop ($\leq$0.25%). We found that the exact accuracy threshold did not have significant impact on the results—see Appendix Figures 17-20 for the full accuracy-effectiveness trade-off.

[2]Note that some of these cases might not be suitable for editing, i.e., when the transformed concept is critical for recognizing the label of an input image (e.g., transforming concept "dog" in images of class "poodle") . We thus manually exclude such concept-class pairs from our analysis—cf. Appendix A.6.1.



(a) ImageNet-trained VGG16 (concepts from COCO) (b) Places-trained ResNet-18 (concepts from LVIS)

Figure 5: Editing vs. fine-tuning, averaged over concept-style pairs. We find that both methods (and their variants) are fairly successful at correcting misclassifications on the target class (examples of which are used to perform the modification). This holds even when the transformation applied during testing is different from the one present in the train exemplars (e.g., a different texture of "wood"). However, crucially, only the improvements induced by editing generalize to other classes where the transformed concept is present, while fine-tuning fails in this setting—typically, causing more errors than it fixes. See Appendix Figures 12-15 for other experimental settings.

4.2 The effectiveness of editing

Recall that we want our prediction-rule rewrites to generalize. That is, if we modify the way that our model treats a specific concept, we want this modification to apply to *every* occurrence of that concept. For instance, if we edit a model to enforce that "wooden wheels" should be treated the same as regular "wheels" in the context of "car" images, we want the model to do the same when encountering other vehicles with "wooden wheels". Thus, while analyzing performance in Figure 5, we treat inputs belonging to the (target) class used to perform the modification separately.

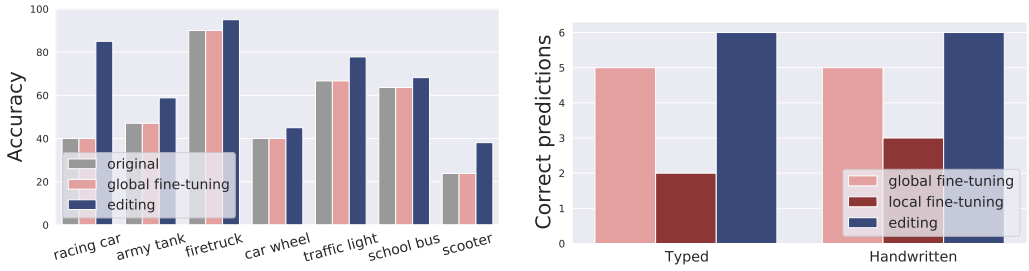
Editing. We find that editing is able to consistently correct mistakes in a manner that *generalizes across classes*. That is, editing is able to reduce errors in non-target classes by often more than 20 percentage points, even though it is performed using only three exemplars from the target class. In Appendix B.2.3, we conduct ablation studies to get a better sense of the key algorithmic factors driving performance. Notably, we find imposing the editing constraints (1) on the entirety of the image—as opposed to only focusing on key-value pairs that correspond to the concept of interest—leads to even better performance (cf. '-mask' in Figure 5). We hypothesize that this has a regularizing effect as it constrains the weights to preserve the original mapping between keys and values in regions that do not contain the concept.

Fine-tuning. In contrast, while the two fine-tuning baselines are able to correct mistakes on transformed inputs of the target class used to perform the modification, they typically *decrease* the model's performance on *other* classes—i.e., they cause more errors than they fix. Moreover, even when we allow a larger drop in the model's accuracy, or use more training exemplars, their performance often becomes *worse* on inputs from other classes—cf. Appendix Figures 17-20. This suggests that fine-tuning causes the model to overfit to the class it was trained on.

Interestingly, we find that in all cases where a method improves performance, this improvement extends to transformations using other variants of the style. For instance, the modification generalizes to textures of "wood" other than those present in exemplars used to perform the modification (cf. Figure 4b). We present examples of errors (not) corrected by editing and fine-tuning in Appendix Figure 16, and provide a per-concept/style break down in Appendix Figures 21 and 22.

5 Real-World Demonstrations

So far, we have seen that our rule rewriting methodology can significantly improve model generalization to concept-level transformations synthesized using our prediction rule-discovery pipeline (cf Section 2). To test the versatility of our approach, we now shift our attention to real-world applications of machine learning models, where similar generalization might be desirable.



(a) Vehicles in snowy weather

(b) Typographic attacks

Figure 6: (a) Adapting a pre-trained ImageNet classifier to images of vehicles on snowy roads with a single exemplar. Fine-tuning (both local and global) only slightly improves accuracy, while editing to map "snowy road"→"road" leads to a consistent improvement across multiple classes. (b) Improving the robustness of CLIP [50] models to typographic attacks [26]. Editing the model to map the text "iPod"→"blank" using a single exemplar—either based on hand-written text on a physical teapot or from pasting typed text on an image of a "can opener"—completely corrects this vulnerability. While global fine-tuning can also improve model performance in this setting, it requires more careful hyperparameter tuning and typically hurts model performance in other contexts.

Tackling new environments: Vehicles on snow. Our first use-case is adapting pre-trained classifiers to image subpopulations that are under-represented in the training data. Specifically, we will focus on the task of recognizing vehicles under heavy snow conditions—a setting that could be pertinent to self-driving cars. To study this problem, we collect a set of real photographs from road-related ImageNet classes using Flickr (details in Appendix A.7). To improve model performance under these conditions, we rewrite its prediction rules: to map "snowy roads" to "road". To do so, we first create a *single synthetic* exemplar: by manually annotating the concept "road" in an ImageNet image from a *different* class (here, "police van"), and transforming it using a snow image obtained from Flickr. We then apply our editing methodology (cf. Section 3), using this single snow-to-road exemplar—see Figure 1.

In Figure 6a, we measure the error rate of the model on the new test set (vehicles in snow) before and after performing the rewrite. We find that our edits significantly improve the model's error rate on these images, despite the fact that we did not use any real "snowy road" photographs to edit the model. In contrast, fine-tuning the model under the same setup does not improve its performance.

Ignoring a spurious feature: Typographic attacks. Our second use-case is modifying a model to ignore a spurious feature. We focus on the recently-discovered typographic attacks from Goh et al. [26]: simply attaching a piece of paper with the text "iPod" on it is enough to make a zero-shot CLIP [50] classifier predict an assortment of other objects to be iPods. We start by reproducing these attacks—see Appendix Figure 8 for an illustration. We now rewrite the model's prediction rules: to map the text "iPod" to "blank" (as the latter does not cause misclassifications). For the choice of our transformed exemplar x', we consider two variants: either a real photograph of a "teapot" with the typographic attack (Appendix Figure 8); or an ImageNet image from of a "can opener" (randomly-chosen) with the typed text "iPod" programatically pasted on it (Figure 1). The original image x for our approach is then obtained by replacing the handwritten/typed text with a white mask—cf. Figure 1. We then use this single training exemplar to perform the network modification.

In both cases, we find that editing (Section 3) is able to fix *all* the errors caused by the typographic attacks, see Figure 6b. Interestingly, global fine-tuning also helps to correct many of these errors (potentially by adjusting class biases), albeit less reliably (for specific hyperparameters). However, unlike editing, fine-tuning also ends up damaging the model behavior in other scenarios—causing it to now spuriously associate the text "iPod" with the target class used for training or significantly reducing the accuracy on normal "iPod" images from the test set (Appendix Figure 29).

6 Related Work

Our work builds upon previous research in interpretability, robustness, and domain adaptation.

Model interpretability. There has been increasing interest in explaining the inner workings of deep classifiers: e.g., by identifying individual neurons neurons [17, 70, 45, 4, 15] or activation vectors [37, 74] that correspond to human-understandable features. The effect of these features on model predictions can be analyzed by either inspecting the model's downstream weights [46, 67] or through counterfactual tests [28, 29, 6]. Here we generate counterfactual images based on pre-trained instance segmentation. This allows us directly measure how individual concepts impact model predictions on large datasets without the need for detailed annotations or human inspection.

Ignoring spurious features. Prior work on preventing models from relying on spurious correlations is based on constraining model predictions to satisfy certain invariances. This can be done by: creating counterfactuals that add or remove objects from scenes [59, 60, 1], learning representations that are simultaneously optimal across domains [2], ensuring comparable performance across data subpopulations [56], or enforcing consistency in predictions across inputs that depict the same physical entity [32]. In this work, we take an alternative approach that does not rely on collecting new inputs or annotations, and instead directly rewrites the model's prediction rules.

Model robustness. A long line of work has been devoted to discovering and correcting failure modes of models. These studies focus on simulating variations in testing conditions that can arise during deployment, including: adversarial or natural input corruptions [63, 18, 19, 16, 20, 33, 35], changes in the data collection process [55, 65, 36, 64, 51], or variations in the data subpopulations present [7, 47, 56, 57, 38]. Typical approaches for improving robustness in these contexts include robust optimization [43, 69, 56] and additional data augmentation [42, 34, 72]. Our rule-discovery and editing pipelines can be viewed as complementary to this work as they allows us to preemptively adjust the model's prediction rules in anticipation of deployment conditions.

Domain adaptation. The goal of domain adaptation is to adapt a model to a specific deployment environment using (potentially unlabeled) samples from it. This is typically achieved by either fine-tuning the model on the new domain [14, 58, 39], by learning the correspondence between the source and target domain, often in a latent representation space [8, 55, 21, 12, 27], or by updating the model's batch normalization statistics [40, 11]. These approaches all require a non-trivial amount of data from the target domain. The question of adaptation from a handful of samples has been explored [44], but in a setting that requires samples across all target classes. In contrast, our method allows for generalization to new (potentially unknown) classes with even a single example.

7 Conclusion

In this work, we developed a general toolkit for performing targeted post-hoc modifications to vision classifiers. Crucially, instead of specifying the desired behavior *implicitly* via training data, our method allows users to *directly* edit the prediction rules learned by models. By doing so, our approach makes it easier for users to encode their prior knowledge and preferences into the model debugging process. Additionally, a key benefit of this technique is that it fundamentally changes how the model processes a given concept—thus making it possible to edit its behavior beyond the specific class(es) used for editing. Finally, our edits do not require any additional data collection: they can be guided by as few as a single (synthetically-created) exemplar. We believe that this primitive opens up new avenues to interact with and correct our models before or during deployment.

Furthermore, the pipeline we designed to discover prediction rules enables one to pinpoint (potentially spurious) high-level concepts in the data that influence model predictions. We thus believe that this pipeline may find additional uses in the context of model interpretability and robustness.

Limitations and broader impact. Even though our methodology is general, performing model edits does require some manual intervention and domain expertise. Further, direct model editing is a departure from the standard way in which machine learned models are trained, and may have broader implications. While we have shown how it can be used to cause beneficial changes in large-scale pre-trained models, direct control of prediction rules could also make it easier for adversaries to introduce vulnerabilities into the model. Overall, direct model editing makes it clearer than ever that our models are a reflection of the goals and biases of we who create them—not only through the training tasks we choose, but now also through the rules that we rewrite.

References

[1] V. Agarwal, R. Shetty, and M. Fritz. Towards causal vqa: Revealing and reducing spurious correlations by invariant and covariant semantic editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9690–9698, 2020.

[2] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[3] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Neural Information Processing Systems (NeurIPS)*, 2019.

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

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

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

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

[8] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Neural Information Processing Systems (NeurIPS)*, 2007.

[9] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. v. d. Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

[10] A. Bissoto, E. Valle, and S. Avila. Debiasing skin lesion datasets and models? not so fast. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 740–741, 2020.

[11] C. Burns and J. Steinhardt. Limitations of post-hoc feature alignment for robustness. 2021.

[12] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. In *Transactions on Pattern Analysis and Machine Intelligence*, 2016.

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

[14] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning (ICML)*, 2014.

[15] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, B. Tran, and A. Madry. Adversarial robustness as a prior for learned representations. In *ArXiv preprint arXiv:1906.00945*, 2019.

[16] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning (ICML)*, 2019.

[17] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. 2009.

[18] A. Fawzi and P. Frossard. Manitest: Are classifiers really invariant? In *British Machine Vision Conference (BMVC)*, 2015.

[19] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[20] N. Ford, J. Gilmer, N. Carlini, and D. Cubuk. Adversarial examples are a natural consequence of test error in noise. In *arXiv preprint arXiv:1901.10513*, 2019.

[21] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. 2015.

[22] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *computer vision and pattern recognition (CVPR)*, 2016.

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

[24] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017.

[25] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

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

[27] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf. Domain adaptation with conditional transferable components. In *International conference on machine learning (ICML)*, 2016.

[28] Y. Goyal, A. Feder, U. Shalit, and B. Kim. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*, 2019.

[29] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee. Counterfactual visual explanations. *arXiv preprint arXiv:1904.07451*, 2019.

[30] A. Gupta, P. Dollar, and R. Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[32] C. Heinze-Deml and N. Meinshausen. Conditional variance penalties and domain shift robustness. *arXiv preprint arXiv:1710.11469*, 2017.

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

[34] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[35] D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt. Testing robustness against unforeseen adversaries. In *ArXiv preprint arxiv:1908.08016*, 2019.

[36] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision (ECCV)*, 2012.

[37] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning (ICML)*, 2018.

[38] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, S. Beery, et al. Wilds: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*, 2020.

[39] A. Kumar, T. Ma, and P. Liang. Understanding self-training for gradual domain adaptation. In *International Conference on Machine Learning (ICML)*, 2020.

[40] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[41] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision (ECCV)*, 2014.

[42] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019.

[43] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[44] S. Motiian, Q. Jones, S. M. Iranmanesh, and G. Doretto. Few-shot adversarial domain adaptation. In *NIPS*, 2017.

[45] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. In *Distill*, 2017.

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

[47] Y. Oren, S. Sagawa, T. Hashimoto, and P. Liang. Distributionally robust language modeling. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[48] J. Pearl. Causal inference. *Causality: Objectives and Assessment*, 2010.

[49] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, 2006.

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

[51] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning (ICML)*, 2019.

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

[53] A. Rosenfeld, R. Zemel, and J. K. Tsotsos. The elephant in the room. In *arXiv preprint arXiv:1808.03305*, 2018.

[54] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *International Journal of Computer Vision (IJCV)*, 2015.

[55] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision (ECCV)*, 2010.

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

[57] S. Santurkar, D. Tsipras, and A. Madry. Breeds: Benchmarks for subpopulation shift. In *International Conference on Learning Representations (ICLR)*, 2021.

[58] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *conference on computer vision and pattern recognition (CVPR) workshops*, 2014.

[59] R. Shetty, M. Fritz, and B. Schiele. Adversarial scene editing: Automatic object removal from weak supervision. In *NeurIPS*, 2018.

[60] R. Shetty, B. Schiele, and M. Fritz. Not using the car to see the sidewalk–quantifying and controlling the effects of context in classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8226, 2019.

[61] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[62] P. Stock and M. Cisse. Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases. In *European Conference on Computer Vision (ECCV)*, 2018.

[63] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[64] T. Tommasi and T. Tuytelaars. A testbed for cross-dataset analysis. In *European Conference on Computer Vision (ECCV)*, 2014.

[65] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, 2011.

[66] D. Tsipras, S. Santurkar, L. Engstrom, A. Ilyas, and A. Madry. From imagenet to image classification: Contextualizing progress on benchmarks. In *International Conference on Machine Learning (ICML)*, 2020.

[67] E. Wong, S. Santurkar, and A. Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning (ICML)*, 2021.

[68] K. Xiao, L. Engstrom, A. Ilyas, and A. Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv preprint arXiv:2006.09994*, 2020.

[69] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer. A fourier perspective on model robustness in computer vision. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[70] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. 2014.

[71] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *International journal of computer vision*, 2007.

[72] L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou. How does mixup help with robustness and generalization? In *International Conference on Learning Representations (ICLR)*, 2021.

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

[74] B. Zhou, Y. Sun, D. Bau, and A. Torralba. Interpretable basis decomposition for visual explanation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See evaluation in Sections 2, 4 and 5.
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Included in the supplementary material as a zip file.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We included bootstrap error bars where applicable.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.3 in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendix A in the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Appendix A in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix A.8 in the supplementary material.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix A.8 in the supplementary material.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]