Leveraging Sparse Linear Layers for Debuggable Deep Networks

Eric Wong [*1] **Shibani Santurkar** [*1] **Aleksander Mądry** [1]

Abstract

We show how fitting sparse linear models over learned deep feature representations can lead to more debuggable deep networks. These networks remain highly accurate while also being more amenable to human interpretation, as we demonstrate quantitatively via numerical and human experiments. We further illustrate how the resulting sparse explanations can help to identify spurious correlations, explain misclassifications, and diagnose model biases in vision and language tasks.[1]

1. Introduction

As machine learning (ML) models find wide-spread application, there is a growing demand for interpretability: access to tools that help people see *why* the model made its decision. There are still many obstacles towards achieving this goal though, particularly in the context of deep learning. These obstacles stem from the scale of modern deep networks, as well as the complexity of even defining and assessing the (often context-dependent) desiderata of interpretability.

Existing work on deep network interpretability has largely approached this problem from two perspectives. The first seeks to uncover the concepts associated with specific neurons in the network, for example via visualization (Yosinski et al., 2015) or semantic labeling (Bau et al., 2017). The second aims to explain model decisions on a per-example basis, using techniques such as local surrogates (Ribeiro et al., 2016a) and saliency maps (Simonyan et al., 2013). While both families of approaches can improve model understanding at a local level—i.e., for a given example or neuron—recent work has argued that such localized explanations can lead to misleading conclusions about the model's overall decision process (Adebayo et al., 2018; 2020; Leavitt & Morcos, 2020). As a result, it is often challenging to flag a model's failure modes or evaluate corrective interventions without in-depth problem-specific studies.

To make progress on this front, we focus on a more actionable intermediate goal of interpretability: *model debugging*. Specifically, instead of directly aiming for a complete characterization of the model's decision process, our objective is to develop tools that help model designers uncover unexpected model behaviors (semi-)automatically.

Our contributions. Our approach to model debugging is based on a natural view of a deep network as the composition of a "deep feature extractor" and a linear "decision layer". Embracing this perspective allows us to focus our attention on probing how deep features are (linearly) combined by the decision layer to make predictions. Even with this simplification, probing current deep networks can be intractable given the large number of parameters in their decision layers. To overcome this challenge, we replace the standard (typically dense) decision layer of a deep network with a sparse but comparably accurate counterpart. This simple approach ends up being surprisingly effective for building deep networks that are intrinsically more debuggable. Specifically, for a variety of modern ML settings:

- We demonstrate that it is possible to construct deep networks that have sparse decision layers (e.g., with only 20-30 deep features per class for ImageNet) without sacrificing much model performance. This involves developing a custom solver for fitting elastic net regularized linear models in order to perform effective sparsification at deep-learning scales.[2]
- We show that sparsifying a network's decision layer can indeed help humans understand the resulting models better. For example, untrained annotators can intuit (simulate) the predictions of a model with a sparse decision layer with high ($\sim$63%) accuracy. This is in contrast to their near chance performance ($\sim$33%) for

[*]Equal contribution [1]Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Correspondence to: Eric Wong <wongeric@mit.edu>, Shibani Santurkar <shibani@mit.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[1]The code for our toolkit can be found at `https://github.com/madrylab/debuggabledeepnetworks`.

[2]A standalone package of our solver is available at `https://github.com/madrylab/glm_saga`

models with standard (dense) decision layers.

- We explore the use of sparse decision layers in three debugging tasks: diagnosing biases and spurious correlations (Section 4.1), counterfactual generation (Section 4.2) and identifying data patterns that cause misclassifications (Section 4.3). To enable this analysis, we design a suite of human-in-the-loop experiments.

2. Debuggability via Sparse Linearity

Recent studies have raised concerns about how deep networks make decisions (Beery et al., 2018; Xiao et al., 2020; Tsipras et al., 2020; Bissoto et al., 2020). For instance, it was noted that skin-lesion detectors rely on spurious visual artifacts (Bissoto et al., 2020) and comment flagging systems use identity group information to detect toxicity (Borkan et al., 2019). So far, most of these discoveries were made via in-depth studies by experts. However, as deep learning makes inroads into new fields, there is a strong case to be made for general-purpose model debugging tools.

While simple models (e.g., small decision trees or linear classifiers) can be directly examined, a similar analysis for typical deep networks is infeasible. To tackle this problem, we choose to decompose a deep network into: (1) a deep feature representation and (2) a linear decision layer. Then, we can attempt to gain insight into the model's reasoning process by directly examining the deep features, and the linear coefficients used to aggregate them. At a high level, our hope is that this decomposition will allow us to get the best of both worlds: the predictive power of learned deep features, and the ease of understanding linear models.

That being said, this simplified problem is still intractable for current deep networks, since their decision layers can easily have millions of parameters operating on thousands of deep features. To mitigate this issue, we instead combine the feature representation of a pre-trained network with a *sparse* linear decision layer (cf. Figure 1). Debugging this sparse decision layer then entails inspecting only the few linear coefficients and deep features that dictate its predictions.

2.1. Constructing sparse decision layers

One possible approach for constructing sparse decision layers is to apply pruning methods from deep learning (LeCun et al., 1990; Han et al., 2015; Hassibi & Stork, 1993; Li et al., 2016; Han et al., 2016; Blalock et al., 2020)—commonly-used to compress deep networks and speed up inference—solely to the dense decision layer. It turns out however that for linear classifiers we can actually do better. In particular, the problem of fitting sparse linear models has been extensively studied in statistics, leading to a suite of methods with theoretical optimality guarantees. These include LASSO regression (Tibshirani, 1994), least angle

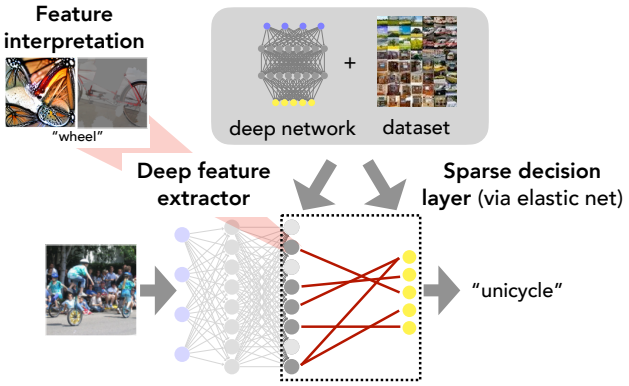


Figure 1: Illustration of our pipeline: For a given task, we construct a *sparse decision layer* by training a regularized generalized linear model (via elastic net) on the deep feature representations of a pre-trained deep network. We then aim to debug model behavior by simply inspecting the few relevant deep features (with existing feature interpretation tools), and the linear coefficients used to aggregate them.

regression (Efron et al., 2004), and forward stagewise regression (Hastie et al., 2007). In this work, we leverage the classic elastic net formulation (Zou & Hastie, 2005)—a generalization of LASSO and ridge regression that addresses their corresponding drawbacks (detailed in Appendix A).

For simplicity, we present an overview of the elastic net for linear regression, and defer the reader to Friedman et al. (2010) for a more complete presentation on the generalized linear model (GLM) in the classification setting. Let (X, y) be the standardized data matrix (mean zero and variance one) and output respectively. In our setting, X corresponds to the (normalized) deep feature representations of input data points, while y is the target. Our goal is to fit a sparse linear model of the form $\mathbb{E}(Y|X = x) = x^T\beta + \beta_0$. Then, the elastic net is the following convex optimization problem:

$$\min_{\beta} \frac{1}{2N}\|X^T\beta + \beta_0 - y\|_2^2 + \lambda R_\alpha(\beta) \tag{1}$$

where

$$R_\alpha(\beta) = (1-\alpha)\frac{1}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1 \tag{2}$$

is referred to as the elastic net penalty (Zou & Hastie, 2005) for given hyperparameters λ and α. Typical elastic net solvers optimize (1) for a variety of regularization strengths $\lambda_1 > \cdots > \lambda_k$, resulting in a series of linear classifiers with weights $\beta_1, \ldots, \beta_k$ known as the *regularization path*, where

$$\beta_i = \arg\min_{\beta} \frac{1}{2N}\|X^T\beta - y\|_2^2 + \lambda_i R_\alpha(\beta) \tag{3}$$

In particular, a path algorithm for the elastic net calculates the regularization path where sparsity ranges the entire spectrum from the trivial zero model ($\beta = 0$) to completely

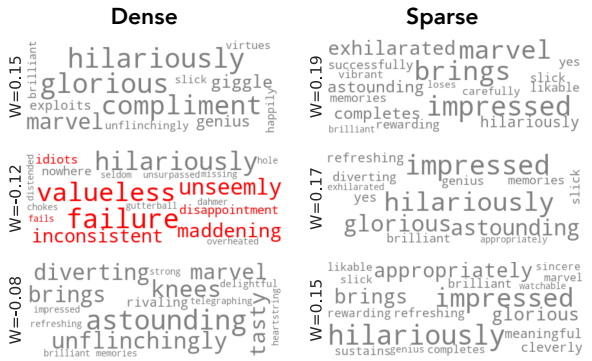


Figure 2: LIME-based word cloud visualizations for the highest weighted features in the (dense/sparse) decision layers of BERT models for *positive* sentiment detection in the SST dataset. As highlighted in red, some of the key features used by the dense decision layer are actually activated for words with *negative* semantic meaning.

dense. This regularization path can then be used to select a single linear model to satisfy application-specific sparsity or accuracy thresholds (as measured on a validation set). In addition, these paths can be used to visualize the evolution of weights assigned to specific features as a function of sparsity constraints on the model, thereby providing further insight into the relative importance of features (cf. Appendix A.3).

Scalable solver for large-scale elastic net. Although the elastic net is widely-used for small-scale GLM problems, existing solvers can not handle the scale (number of samples and input dimensions) that typically arise in deep learning. In fact, at such scales, state-of-the-art solvers struggle to solve the elastic net even for a single regularization value, and cannot be directly parallelized due to their reliance on coordinate descent (Friedman et al., 2010). We remedy this by creating an optimized GLM solver that combines the path algorithm of Friedman et al. (2010) with recent advancements in variance reduced gradient methods (Gazagnadou et al., 2019). The speedup in our approach comes from the improved convergence rates of these methods over stochastic gradient descent in strongly convex settings such as the elastic net. Using our approach, we can fit ImageNet-scale regularization paths to numerical precision on the order of hours on a single GPU (cf. Appendix A.1 for details).

2.2. Interpreting deep features

A sparse linear model allows us to reason about the network's decisions in terms of a significantly smaller set of deep features. When used in tandem with off-the-shelf feature interpretation methods, the end result is a simplified description of how the network makes predictions. For our study, we utilize the following two widely-used techniques:

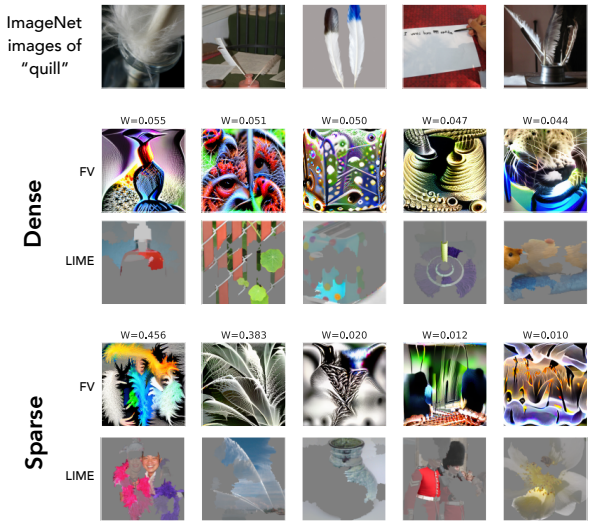


Figure 3: Visualization of deep features used by dense and sparse decision layers of a robust ($\varepsilon = 3$) ResNet-50 classifier to detect the ImageNet class "quill". Here we present five deep features used by each decision layer, that are randomly-chosen from the top-k highest-weighted ones—where k is the number of features used by the sparse decision layer for this class. For each (deep) feature, we show its linear coefficient (W), feature visualization (FV) and LIME superpixels.

1. *LIME (Ribeiro et al., 2016a)*: Although traditionally used to interpret model outputs, we use it to understand deep features. We fit a local surrogate model around the most activating examples of a deep feature to identify key "superpixels" for images or words for sentences.

2. *Feature visualization (Yosinski et al., 2015)*: Synthesizes inputs that maximally activate a given neuron.[3]

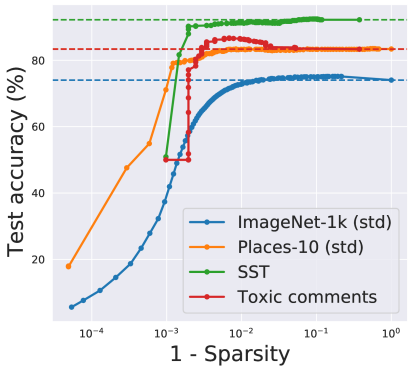
We detail the visualization procedure in Appendix B, and present sample visualizations in Figure 2 and Figure 3.

3. Are Sparse Decision Layers Better?

We now apply our methodology to widely-used deep networks and assess the quality of the resulting sparse decision layers along a number of axes. We demonstrate that:

1. The standard (henceforth referred to as *"dense"*) linear decision layer can be made highly sparse at only a

[3] Despite significant research, feature visualizations for standard vision models are often hard to parse, possibly due to their reliance on human-unintelligible features (Ilyas et al., 2019). Thus, in the main paper, we present visualizations from adversarially-trained models which tend to have more human-aligned features (Tsipras et al., 2019; Engstrom et al., 2019), and present the corresponding plots for standard models in Appendix D.3.



(a)

Dataset/Model	k	Dense			Sparse		
		All	Top-k	Rest	All	Top-k	Rest
ImageNet (std)	10	74.03	58.46	55.22	72.24	69.78	10.84
ImageNet (robust)	10	61.23	28.99	34.65	59.99	45.82	19.83
Places-10 (std)	10	83.30	83.60	81.20	77.40	77.40	10.00
Places-10 (robust)	10	80.20	76.10	76.40	77.80	76.60	40.20
SST	5	91.51	53.10	91.28	90.37	90.37	50.92
Toxic comments	5	83.33	55.35	57.87	82.47	82.33	50.00
Obscene comments	5	80.41	50.03	50.00	77.32	72.39	50.00
Insult comments	5	72.72	50.00	50.00	77.14	75.80	50.00

(b)

Figure 4: (a): Sparsity vs. accuracy trade-offs of sparse decision layers (cf. Appendix Figure 16 for additional models/tasks). Each point on the curve corresponds to single linear classifier from the regularization path in Equation (3). (b): Comparison of the accuracy of dense/sparse decision layers when they are constrained to utilize only the top-k deep features (based on weight magnitude). We also show overall model accuracy, and the accuracy gained by using the remaining deep features.

small cost to performance (Section 3.1).

2. The deep features selected by sparse decision layers are qualitatively and quantitatively better at summarizing the model's decision process (Section 3.2). Note that the dense and sparse decision layers operate on the same deep features—they only differ in the weight (if any) they assign to each one.

3. These aforementioned improvements (induced by the sparse decision layer) translate into better human understanding of the model (Section 3.3).

We perform our analysis on: (a) ResNet-50 classifiers (He et al., 2016) trained on ImageNet-1k (Deng et al., 2009; Russakovsky et al., 2015) and Places-10 (a 10-class subset of Places365 (Zhou et al., 2017)); and (b) BERT (Devlin et al., 2018) for sentiment classification on Stanford Sentiment Treebank (SST) (Socher et al., 2013) and toxicity classification of Wikipedia comments (Wulczyn et al., 2017). Details about the setup can be found in Appendix C.

3.1. Sparsity vs. performance

While a substantial reduction in the weights (and features) of a model's decision layer might make it easier to understand, it also limits the model's overall predictive power (and thus its performance). Still, we find that across datasets and architectures, the decision layer can be made substantially sparser—by up to two orders of magnitude—with a small impact on accuracy (cf. Figure 4a). For instance, it is possible to find an accurate decision layer that relies on only about 20 deep features/class for ImageNet (as opposed to 2048 in the dense case). Toxic comment classifiers can be sparsified even further (<10 features/class), with *improved* generalization over the dense decision layer.

For the rest of our study, we select a single sparse decision layer to balance performance and sparsity—specifically the sparsest model whose accuracy is within 5% of top validation set performance (details in Appendix D.1.1). However, as discussed previously, these thresholds can be varied based on the needs of specific applications.

3.2. Sparsity and feature highlighting

Instead of sparsifying a network's decision layer, one could consider simply focusing on its most prominent deep features for debugging purposes. In fact, this is the basis of feature highlighting or principal reason explanations in the credit industry (Barocas et al., 2020). How effective are such feature highlighting explanations at mirroring the underlying model?

In Table 4b, we measure the accuracy of the dense/sparse decision layer when it is constrained to utilize only the top-k (5-10) features by weight magnitude. For dense decision layers, we consistently find that the top-k features do not fully capture the model's performance. This is in stark contrast to the sparse case, where the top-k features are both necessary, and to a large extent sufficient, to capture the model's predictive behavior. Note that the top-k features of the dense decision layers in the language setting almost completely fail at near random-chance performance ($\sim$50%). This indicates that there do exist cases where focusing on the most important features (by weight) of a dense decision layer provides a misleading picture of global model behavior.

Table 5: Bias detection in language models: using sparse decision layers, we find that Debiased-BERT is *still* disproportionately sensitive to identity groups—except that it now uses this information as evidence against toxicity. For example, simply adding the word "christianity" to clearly toxic sentences flips the prediction of the model to non-toxic (score < 0.5).

Toxic sentence	Change in score
DJ Robinsin is [redacted]! he [redacted] so much! [+christianity]	$0.52 \rightarrow 0.49$
Jeez Ed, you seem like a [redacted] [+christianity]	$0.52 \rightarrow 0.48$
Hey [redacted], quit removing FACTS from the article [redacted]!! [+christianity]	$0.51 \rightarrow 0.45$

3.3. Sparsity and human understanding

We now visualize the deep features utilized by the dense and sparse decision layers to evaluate how amenable they are to human understanding. We show representative examples from sentiment classification (SST) and ImageNet, and provide additional visualizations in Appendix D.3.

Specifically, in Figure 2, we present word cloud interpretations of the top three deep features used by both of these decision layers for detecting positive sentiment on the SST dataset (Socher et al., 2013). It is apparent that the sparse decision layer selects features which activate for words with positive semantic meaning. In contrast, the second most prominent deep feature for the dense decision layer is actually activated by words with *negative* semantic meaning. This example highlights how the dense decision layer can lead to unexpected features being used for predictions.

In Figure 3, we present feature interpretations corresponding to the ImageNet class "quill" for both the dense and sparse decision layers of a ResNet-50 classifier[3]. These feature visualizations seem to suggest that the sparse decision layer focuses more on deep features which detect salient class characteristics, such as "feather-like texture" and the "glass bottle" in the background.

Model simulation study To validate the perceived differences in the vision setting—and ensure they are not due to confirmation biases—we conduct a human study on Amazon Mechanical Turk (MTurk). Our goal is to assess how well annotators are able to intuit (simulate[4]) overall model behavior when they are exposed to its decision layer. To this end, we show annotators five randomly-chosen features used by the (dense/sparse) decision layer to recognize objects of a target class, along with the corresponding linear coefficients. We then present them with three samples from the validation set and ask them to choose the one that best matches the target class (cf. Appendix Figure 23 for a sample task). Crucially, annotators are not provided with any information regarding the target class, and must make their prediction based solely on the visualized features.

For both the dense and sparse decision layers, we evaluate how accurate annotators are on average (over 1000 tasks)—based on whether they can correctly identify the image with the highest target class probability according to the corresponding model. For the model with a sparse decision layer, annotators succeed in guessing the predictions in $63.02 \pm 3.02\%$ of the cases. In contrast, they are only able to attain $35.61 \pm 3.09\%$ accuracy—which is near-chance (33.33%)—for the model with a dense decision layer. Crucially, these results hold *regardless* of whether the correct image is actually from the target class or not (see Appendix Table 25 for a discussion).

Note that our task setup precludes annotators from succeeding based on any prior knowledge or cognitive biases as we do not provide them with any semantic information about the target label, aside from the feature visualizations. Thus, annotators' success on this task in the sparse setting indicates that the sparse decision layer is actually effective at reflecting the model's internal reasoning process.

4. Debugging deep networks

We now demonstrate how deep networks with sparse decision layers can be substantially easier to debug than their dense counterparts. We focus on three problems: detecting biases, creating counterfactuals, and identifying input patterns responsible for misclassifications.

4.1. Biases and (spurious) correlations

Our first debugging task is to automatically identify unintended biases or correlations that deep networks extract from their training data.

Toxic comments. We start by examining two BERT models trained to classify comments according to toxicity: (1) Toxic-BERT, a high-performing model that was later found to use identity groups as evidence for toxicity, and (2) Debiased-BERT, which was trained to mitigate this bias (Hanu & Unitary team, 2020).

[4]Simulatibility is a standard evaluation metric in interpretability (Ribeiro et al., 2016b; Lipton, 2018), wherein an explanation is deemed good if it enables humans to reproduce what the model will decide (irrespective of the "correctness" of said decision).

We find that Toxic-BERT models with sparse decision layers also rely on identity groups to predict comment toxicity (visualizations in Appendix E.1 are censored). Words related to nationalities, religions, and sexual identities that are not inherently toxic occur frequently and prominently, and comprise 27% of the word clouds shown for features that detect toxicity. Note that although the standard Toxic-BERT model is known to be biased, this bias is not as apparent in the deep features used by its (dense) decision layer (cf. Appendix E.1). In fact, measuring the bias in the standard model required collecting identity and demographic-based subgroup labels (Borkan et al., 2019).

We can similarly inspect the word clouds for the Debiased-BERT model with sparse decision layers and corroborate that identity-related words no longer appear as evidence for toxicity. But rather than ignoring these words completely, it turns out that this model uses them as strong evidence *against* toxicity. For example, identity words comprise 43% of the word clouds of features detecting non-toxicity. This suggests that the debiasing intervention proposed in Borkan et al. (2019) may not have had the intended effect—Debiased-BERT is still disproportionately sensitive to identity groups, albeit in the opposite way.

We confirm that this is an issue with Debiased-BERT via a simple experiment: we take toxic sentences that this model (with a sparse decision layer) correctly labels as toxic, and simply append an identity related word (as suggested by our word clouds) to the end—see Table 5. This modification turns out to strongly impact model predictions: for example, just adding "christianity" to the end of toxic sentences flips the prediction to non-toxic 74.4% of the time. We note that the biases diagnosed via sparse decision layers are also relevant for the standard Debiased-BERT model. In particular, the same toxic sentences with the word "christianity" are classified as non-toxic 62.2% of the time by the standard model, even though this sensitivity is not as readily apparent from inspecting its decision layer (cf. Appendix E.1).

ImageNet. We now move to the vision setting, with the goal of detecting spurious feature dependencies in ImageNet classifiers. Once again, our approach is based on the following observation: input-class correlations learned by a model can be described as the data patterns (e.g., "dog ears" or "snow") that activate deep features used to recognize objects of that class, according to the decision layer.

Even so, it is not clear how to identify such patterns for image data, without access to fine-grained annotations describing image content. To this end, we rely on a human-in-the-loop approach (via MTurk). Specifically, for a deep feature of interest—used by the sparse decision layer to detect a target class—annotators are shown examples of images that activate it. Annotators are then asked if these "prototypical" images have a shared visual pattern, and if so, to describe it using free-text.

However, under this setup, presenting annotators with images from the target class alone can be problematic. After all, these images are likely to have multiple visual patterns in common—not all of which cause the deep feature to activate. Thus, to disentangle the pertinent data pattern, we present annotators with prototypical images drawn from more than one classes. A sample task is presented in Appendix Figure 28, wherein annotators see three highly-activating images for a specific deep feature from two different classes, along with the respective class labels. Aside from asking annotators to validate (and describe) the presence of a shared pattern between these images, we also ask them whether the pattern (if present) is part of each class object (non-spurious correlation) or its surroundings (spurious correlation)[5].

Table 6: The percentage of class-level correlations identified using our MTurk setup, along with a breakdown of whether annotators believe the pattern to be "non-spurious" (i.e., part of the object) or "spurious" (i.e., part of the surroundings).

Detected patterns (%)	Dense	Sparse
Non-spurious	18.43 ± 2.48	34.43 ± 3.38
Spurious	9.56 ± 1.76	12.49 ± 2.02
Total	27.85 ± 2.70	46.97 ± 3.15

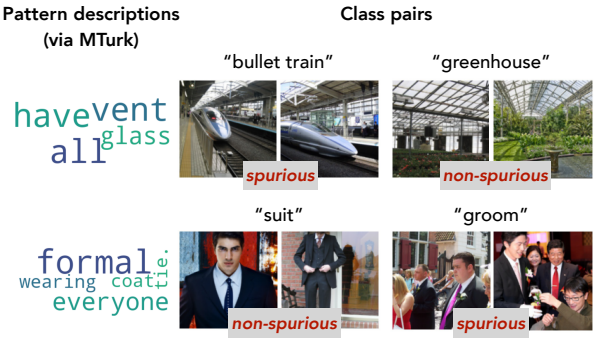


Figure 7: Examples of correlations in ImageNet models detected using our MTurk study. Each row contains protypical images from a pair of classes, along with the annotator-provided descriptions for the shared deep feature that these images strongly activate. For each class, we also display if annotators marked the feature to be a "spurious correlation".

We find that annotators are able to identify a significant number of correlations that standard ImageNet classifiers rely on (cf. Table 6). Once again, sparsity seems to aids the detection of such correlations. Aside from having fewer (deep)

[5]We focus on this specific notion of "spurious correlations" as it is easy for humans to verify—cf. Appendix E.2 for details.

(a)

Original sentence	Counterfactual	Change in score
...something *likable* about the marquis...	...something *irritating* about the marquis...	$0.73 \rightarrow 0.34$
Slick piece of cross-promotion	*Hype* piece of cross-promotion	$0.73 \rightarrow 0.34$
A *marvel* like none you've seen	A *failure* like none you've seen	$0.73 \rightarrow 0.31$

(b)

Figure 8: (a): Word cloud visualization for tokens that are positively/negatively correlated with the activation of a particular deep feature. (b): Using the wordclouds from (a), we can make word substitutions (as highlighted in green and red) to generate counterfactuals that change the model's predicted sentiment (scores below 0.5 are predicted as negative).

feature dependencies per class, it turns out that annotators are able to pinpoint the (shared) data patterns that trigger the relevant deep features in 20% more cases for the model with a sparse decision layer. Interestingly, the fraction of detected patterns that annotators deem spurious is lower for the sparse case. In Figure 7, we present examples of detected correlations with annotator-provided descriptions as word clouds (cf. Appendix E.2 for additional examples). A global word cloud visualization of correlations identified by annotators is shown in Appendix Figure 30.

4.2. Counterfactuals

A natural way to probe model behavior is by trying to find small input modifications which cause the model to change its prediction. Such modified inputs, which are (a special case of) *counterfactuals*, can be a useful primitive for pinpointing input features that the model relies on. Aside from debugging, such counterfactuals can also be used to provide users with recourse (Ustun et al., 2019) that can guide them to obtaining better outcomes in the future. We now leverage the deep features used by sparse decision layers to inform counterfactual generation.

Sentiment classifiers. Our goal here is to automatically identify word substitutions that can be made within a sentence to flip the sentiment label assigned by the model. We do this as follows: given a sentence with a positive sentiment prediction, we first identify the set of deep features used by the sparse decision layer that are positively activated for any word in the sentence. For a randomly chosen deep feature from this pool, we then substitute the positive word from the sentence with its negative counterpart. This substitute is in turn randomly chosen from the set of words that negatively activate the same deep feature (based on its word cloud). An example of the positive and negative word clouds for one such deep feature is shown in Figure 8a, and the resulting counterfactuals are in Table 8b (cf. Appendix F for details). Counterfactuals generated in this manner successfully flip the sentiment label assigned by the sparse decision layer $73.1 \pm 3.0\%$ of the time. In contrast, such counterfactuals only have $52.2 \pm 4\%$ efficacy for the dense decision layer. This highlights that for models with sparse decision layers, it can be easier to automatically identify deep features that are causally-linked to model predictions.

ImageNet. We now leverage the annotations collected in Section 4.1 to generate counterfactuals for ImageNet classifiers. Concretely, we manually modify images to add or subtract input patterns identified by annotators and verify that they successfully flip the model's prediction. Some representative examples are shown in Figure 9. Here, we alter images from various classes to have the pattern "chain-link fence" and "water", so as to fool the sparse decision layer into recognizing them as "ballplayers" and "snorkels" respectively. We find that we are able to consistently change the prediction of the sparse decision layer (and in some cases its dense counterpart) by adding a pattern that was previously identified to be a spurious correlation.

4.3. Misclassifications

Our final avenue for diagnosing unintended behaviors in models is through their misclassifications. Concretely, given an image for which the model makes an incorrect prediction (i.e., not the ground truth label as per the dataset), our goal is to pinpoint some aspects of the image that led to this error.

In the ImageNet setting, it turns out that over 30% of misclassifications made by the sparse decision layer can be attributed to a single deep feature—i.e., manually setting this "problematic" feature to zero fixes the erroneous prediction. For these cases, can humans understand why the problematic feature was triggered in the first place? Namely, can they recognize the *input pattern* that caused the error?

To test this, we present annotators on MTurk with misclassified images. Without divulging the ground truth or predicted

Figure 9: ImageNet counterfactuals. We manually modify samples (*top row*) to contain the patterns "chainlink fence" and "water", which annotators deem (cf. Section 4.1) to be spuriously correlated with the classes "ballplayer" and "snorkel" respectively. We find that these counterfactuals (*bottom row*) succeed in flipping the prediction of the model with a sparse decision layer to the desired class.

Table 10: Fraction of misclassified images for which annotators select the top feature of the predicted class to: (i) match the given image and (ii) be a better match than the top feature for the ground truth class. As a baseline, we also evaluate annotator selections when the top feature for the predicted class is replaced by a randomly-chosen one.

Features	Matches image	Best match
Prediction	70.70% ± 3.62%	60.12% ± 3.77%
Random	16.63% ± 2.91%	10.58% ± 2.35%

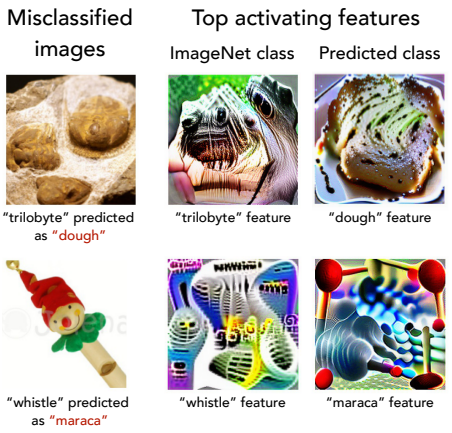


Figure 11: Examples of misclassified ImageNet images for which annotators deem the top activated feature for the predicted class (*rightmost*) as a better match than the top activated feature for the ground truth class (*middle*).

labels, we show annotators the top activated feature for each of the two classes via feature visualizations. We then ask annotators to select the patterns (i.e., feature visualizations) that match the image, and to choose one that is a better match for the image (cf. Appendix G.1 for details). As a control, we repeat the same task but replace the problematic feature with a randomly-chosen one.

For about 70% of the misclassified images, annotators select the top feature for the predicted class as being present in the image (cf. Table 10). In fact, annotators consider it a better match than the feature for the ground truth class 60% of the time. In contrast, they rarely select randomly-chosen features to be present in the image. Since annotators do not know what the underlying classes are, the high fraction of selections for the problematic feature indicates that annotators actually believe this pattern is present in the image.

We present sample misclassifications validated by annotators in Figure 11, along with the problematic features that led to them. Having access to this information can guide improvements in both models and datasets. For instance, model designers might consider augmenting the training data with examples of "maracas" without "red tips" to correct the second error in Figure 11. In Appendix G.3, we further discuss how sparse decision layers can provide insight into inter-class model confusion matrices.

5. Related Work

We now discuss prior work in interpretability and generalized linear models. Due to the large body of work in both fields, we limit the discussion to closely-related studies.

Interpretability tools. There have been extensive efforts towards post-hoc interpretability tools for deep networks. Feature attribution methods provide insight into model predictions for a specific input instance. These include saliency maps (Simonyan et al., 2013; Smilkov et al., 2017; Sundararajan et al., 2017), surrogate models to interpret local decision boundaries (Ribeiro et al., 2016a), and finding influential (Koh & Liang, 2017), prototypical (Kim et al., 2016), or counterfactual inputs (Goyal et al., 2019b). However, as noted by various recent studies, these local attributions can be easy to fool (Ghorbani et al., 2019a; Slack et al., 2020) or may otherwise fail to capture global aspects of model behavior (Sundararajan et al., 2017; Adebayo et al., 2018; 2020; Leavitt & Morcos, 2020). Several methods have been proposed to interpret hidden units within vision networks, for example by generating feature visualizations (Erhan et al., 2009; Yosinski et al., 2015; Nguyen et al., 2016; Olah et al., 2017) or assigning semantic concepts to them (Bau et al., 2017; 2020). Our work is complementary to these methods as we use them as primitives to probe sparse decision layers. Another related line of work is that on concept-based explanations, which seeks to explain the behavior of deep networks in terms of high-level concepts (Kim et al., 2018;

Ghorbani et al., 2019b; Yeh et al., 2020). One of the drawbacks of these methods is that the detected concepts need not be causally linked to the model's predictions (Goyal et al., 2019a). In contrast, in our approach, the identified high-level concepts, i.e., the deep features used by the sparse decision layer, entirely determine the model's behavior.

Most similar is the recent work by (Wan et al., 2020), which proposes fitting a decision tree on a deep feature representation. Network decisions are then explained in terms of semantic descriptions for nodes along the decision path. Wan et al. (2020) rely on heuristics for fitting and labeling the decision tree, that require an existing domain-specific hierarchy (e.g., WordNet), causing it to be more involved and limited in its applicability than our approach.

Regularized GLMs and gradient methods. Estimating GLMs with convex penalties has been studied extensively. Algorithms for efficiently computing regularization paths include least angle regression for LASSO (Efron et al., 2004) and path following algorithms (Park & Hastie, 2007) for ℓ_1 regularized GLMs. The widely-used R package `glmnet` by Friedman et al. (2010) provides an efficient coordinate descent-based solver for GLMs with elastic net regularization, and attains state-of-the-art solving times on CPU-based hardware. Unlike our approach, this library is best suited for problems with few examples or features, and is not directly amenable to GPU acceleration. Our solver also builds off a long line of work in variance reduced proximal gradient methods (Johnson & Zhang, 2013; Defazio et al., 2014; Gazagnadou et al., 2019), which have stronger theoretical convergence rates compared to stochastic gradient descent.

6. Conclusion

We demonstrate how fitting sparse linear models over deep representations can result in more debuggable models, and provide a diverse set of scenarios showcasing the usage of this technique in practice. The simplicity of our approach allows it to be broadly applicable to any deep network with a final linear layer, and may find uses beyond the language and vision settings considered in this paper.

Furthermore, we have created a number of human experiments for tasks such as testing model simulatiblity, detecting spurious correlations and validating misclassifications. Although primarily used in the context of evaluating the sparse decision layer, the design of these experiments may be of independent interest.

Finally, we recognize that while deep networks are popular within machine learning and artifical intelligence settings, linear models continue to be widely used in other scientific fields. We hope that the development and release of our elastic net solver will find broader use in the scientific community for fitting large scale sparse linear models in contexts beyond deep learning.

Acknowledgements

We thank Dimitris Tsipras for helpful discussions.

Work supported in part by the Google PhD Fellowship, Open Philanthropy, and NSF grants CCF-1553428 and CNS-1815221. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0015. Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Adebayo, J., Muelly, M., Liccardi, I., and Kim, B. Debugging tests for model explanations. 2020.

Barocas, S., Selbst, A. D., and Raghavan, M. The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020.

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

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

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

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

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

Borkan, D., Dixon, L., Sorensen, J., Thain, N., and Vasserman, L. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, pp. 491–500, 2019.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems (NeurIPS)*, 2014.

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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. Least angle regression. *The Annals of statistics*, 2004.

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

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

Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 2010.

Gazagnadou, N., Gower, R. M., and Salmon, J. Optimal mini-batch and step sizes for saga. *arXiv preprint arXiv:1902.00071*, 2019.

Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019a.

Ghorbani, A., Wexler, J., Zou, J., and Kim, B. Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*, 2019b.

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

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

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

Hanu, L. and Unitary team. Detoxify. Github. https://github.com/unitaryai/detoxify, 2020.

Hassibi, B. and Stork, D. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, 1993.

Hastie, T., Taylor, J., Tibshirani, R., Walther, G., et al. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29, 2007.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

Kim, B., Khanna, R., and Koyejo, O. O. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems (NeurIPS*, 2016.

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

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *ICML*, 2017.

Leavitt, M. L. and Morcos, A. Towards falsifiable interpretability research. *arXiv preprint arXiv:2010.12016*, 2020.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*. Morgan-Kaufmann, 1990.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Lipton, Z. C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. 2018.

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

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Neural Information Processing Systems (NeurIPS)*, 2016.

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

Park, M. Y. and Hastie, T. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2007.

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

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

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

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. SmoothGrad: removing noise by adding noise. In *ICML workshop on visualization for deep learning*, 2017.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

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

Tibshirani, R. Regression shrinkage and selection via the lasso. In *Journal of the Royal Statistical Society, Series B*, 1994.

Tibshirani, R. and Wasserman, L. Sparsity, the lasso, and friends. *Lecture notes from "Statistical Machine Learning," Carnegie Mellon University, Spring*, 2017.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Ustun, B., Spangher, A., and Liu, Y. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

Wan, A., Dunlap, L., Ho, D., Yin, J., Lee, S., Jin, H., Petryk, S., Bargal, S. A., and Gonzalez, J. E. Nbdt: neural-backed decision trees. *arXiv preprint arXiv:2004.00221*, 2020.

Wulczyn, E., Thain, N., and Dixon, L. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 1391–1399, 2017.

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

Yeh, C.-K., Kim, B., Arik, S., Li, C.-L., Pfister, T., and Ravikumar, P. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. In *arXiv preprint arXiv:1506.06579*, 2015.

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

Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 2005.