Meaningfully Explaining a Model's Mistakes

Abubakar Abid
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
a12d@stanford.edu

James Zou
Department of Biomedical Engineering
Stanford University
Stanford, CA 94305
jamesz@stanford.edu

Abstract

Understanding and explaining the mistakes made by trained models is critical to many machine learning objectives, such as improving robustness, addressing concept drift, and mitigating biases. However, this is often an ad hoc process that involves manually looking at the model's mistakes on many test samples and guessing at the underlying reasons for those incorrect predictions. In this paper, we propose a systematic approach, *conceptual explanation scores* (CES), that explains why a classifier makes a mistake on a particular test sample(s) in terms of human-understandable concepts (e.g. this zebra is misclassified as a dog because of faint *stripes*). We base CES on two prior ideas: counterfactual explanations and concept activation vectors, and validate the approach on well-known pretrained models, showing that it explains the models' mistakes meaningfully. We also train new models with intentional and known spurious correlations, which CES successfully identifies from a *single* misclassified test sample, and we further show that we can use these explanations to *correct* mistaken predictions. The code for CES is publicly available and can easily be applied to explain mistakes in new models.

1 Introduction

People who use machine learning (ML) models often need to understand why a trained model is making a particular mistake. For example, upon seeing a model misclassify an image, an ML practitioner may ask questions such as: *Was this kind of image underrepresented in my training distribution? Am I preprocessing the image correctly? Has my model learned a spurious correlation or bias that is hindering generalization?* Answering this question correctly affects the *usability* of a model and can help make the model more *robust*. Consider a motivating example:

Example 1: Usability of a Pretrained Model. A pathologist downloads a pretrained model to classify histopathology images. Despite a high reported accuracy, he finds the model making mistakes on his images. He investigates why that is the case, finding that the ***hues*** in his images are different than in the original training data. Realizing the issue, he is able to transform his own images with some preprocessing, matching the training distribution and improving the model's performance.

In the above example, we have domain shift occurring between training and test time, which degrades the model's performance [Subbaswamy et al., 2019, Koh et al., 2020]. By *explaining* the cause of the domain shift, we are able to easily fix the model's predictions. On the other hand, if the domain shift is due to more complex spurious correlations the model has learned, it might need to be completely *retrained* before it can be used. During development, identifying and explaining a model's failure points can make it more robust [Abid et al., 2019, Kiela et al., 2021], as in the following example:

Example 2: Discovering Biases During Development. A dermatologist trains a machine learning classifier to classify skin diseases from skin images collected from patients at her hospital. She shares her trained model with a colleague at a different hospital, who reports that the model makes mistakes

Preprint. Under review.

with his own patients. She investigates why the model is making mistakes, realizing that patients at her colleague's hospital have different ***skin colors*** and ***ages***. Answering this question allows her to not only know her own model's biases, but also guides her to augment her training set with the right kind of data to build a more robust model.

Explaining a model's mistakes is also useful in other settings where data distributions may change, such as concept drift for deployed machine learning models [Lu et al., 2018]. Despite its usefulness, explaining a model's performance drop is often an ad hoc process that involves manually looking at the model's mistakes on many test samples and guessing at the underlying reasons for those incorrect predictions. In this paper, we present ***conceptual explanation scores*** (CES), a systematic method for explaining model mistakes in terms of meaningful human-understandable concepts, such as those in the examples above (e.g. *hues*). In addition, CES was developed with the following goals:

- **No retraining**: CES does not require model retraining or even access to any training data.
- **Speed**: after a one-time step to learn concepts (which can be performed beforehand), CES can be applied very efficiently to explain a mistake on a specific sample or samples.
- **Generality**: In this paper, we apply CES to image classification, but it can be extended naturally to other model types (see Section 5)

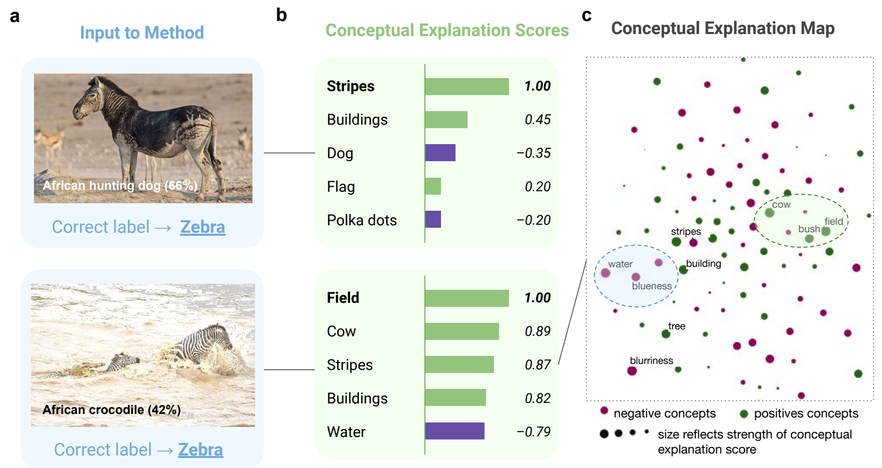

Figure 1: **Explaining with Conceptual Explanation Scores:** **(a)** Given a misclassified sample, such as one of the images shown here (top image misclassified by SqueezeNet [Iandola et al., 2016] as an African hunting dog and the bottom as an African crocodile), and its correct label (zebra), **(b)** our method assigns a conceptual explanation score (CES) between -1 and 1 to a set of predefined concepts. A large positive score means that adding that concept to the image (e.g. *stripes*, top image) will increase the probability of correctly classifying the image, as will removing or reducing a concept with a large negative score (e.g. *water*, bottom image). **(c)** To provide more holistic explanations, a conceptual explanation map (CEM) showing the 2D embeddings of each concept, along with its score can be produced. Here, we generate a CEM for the bottom image and label the top 9 concepts, revealing a cluster with negative scores (blue dashed line) related to water, and a cluster with positives scores (green dashed line) related to grass or land. This leads us to conclude that we can improve the robustness of our model by training our model with more images of zebras in water, not only land.

An overview of our method, as applied to two sample images, is shown in Fig. 1. To summarize **our contributions**, we develop a novel method, CES, that can systematically explain model mistakes in terms of high-level concepts that are easy for users to understand. We show that CES correctly identifies spurious correlations in model training, and we quantitatively validate CES across different experiments. We have released all of the code and data needed for our method and experiments in a public repository: `https://github.com/conceptualexplanations/submission`.

2 Related Works

Our approach is inspired by prior efforts to explain machine learning models' predictions. Two ideas are particularly relevant: *counterfactual explanations* and *concept activation vectors*. Our approach can also improve *model robustness*, particularly with changing data distributions, for which other methods have also been proposed in the literature. Here, we provide an overview of these methods and discuss differences between them and our proposed approach.

2.1 Counterfactual Explanations

Counterfactual explanations (Verma et al. [2020] provides a comprehensive review) are a class of model interpretation methods that seek to answer: *what perturbations to the input are needed for a model's prediction to change in a particular way?* A large number of counterfactual explanation methods have been proposed with various desiderata such as sparse perturbations [Wachter et al., 2017], perturbations that remain close to the data manifold [Dhurandhar et al., 2018], and causality [Mahajan et al., 2019]. What these methods share in common is that they provide an interpretable link between model predictions and perturbations to input features.

To compute CES, we use a similar approach, perturbing input data to change its predictions, but for a complementary goal: to understand the limitations and biases of our model and its training data, rather than to change a specific prediction. Furthermore, we extend these perturbations to unstructured data (specifically, natural images), whereas existing counterfactual explanation methods have been largely confined to tabular datasets since the perturbations have consisted of changing values in the original low-level feature space.

2.2 Concept Activation Vectors

To explain an image classification model's mistakes in a useful way, we need to operate not with low-level features, but with more meaningful concepts. Concept activation vectors (CAVs) [Kim et al., 2018] are a powerful method to understand the internals of a network in human-interpretable concepts. CAVs are linear classifiers trained in the bottleneck layers of a network and correspond to concepts that are usually user-defined or automatically discovered from data [Ghorbani et al., 2019].

In previous literature, CAVs have been used to test the association the between a concept and the model's prediction on a class of training data [Kim et al., 2018] as well as provide visual explanations for predictions [Zhou et al., 2018]. Our approach extends CAVs by showing that *perturbations* along the CAV can be used to change a model's prediction on a specific test sample, e.g. to correct a mistaken prediction, and thereby be used in a similar manner to counterfactuals. Since samples (even of the same class) can be misclassified for different reasons (see Fig. 1), our approach allows a more contextual understanding of a model's behavior and mistakes.

2.3 Model Biases and Robustness

Understanding a model's behavior and explaining its mistakes is critical for building more robust and less biased models, as we have discussed in Section 1. While some other methods, such as FairML [Adebayo and Kagal, 2016] and the What-If Tool [Wexler et al., 2020], can also be used to discover biases and failure points in models, they are typically limited to tabular datasets where sensitive features are explicitly designated, unlike our approach which is designed for unstructured data and works with a more diverse set of concepts.

Our approach can also be used to explain a model's performance degradation with data drift. It is complementary to methods for out-of-distribution sample detection [Hendrycks and Gimpel, 2016] and black-box shift detection [Lipton et al., 2018], which can detect data drift, but cannot explain the underlying reasons. There are also algorithmic approaches to improving model robustness with data drift [Raghunathan et al., 2018, Nestor et al., 2019]. Although these methods can provide some guarantees around robustness for unstructured data, they are in terms of low-level perturbations, and do not hold with more conceptual and natural changes in data distributions.

Algorithm 1: Generating CES for a Misclassified Sample

```
Inputs :
        f  # trained network:  model
        L  # bottleneck layer (hyperparameter):  int
        delta  # step size (hyperparameter):  float
        concepts # set of concepts:  set[str]
        P  # positive examples per concept:  dict[str, list[sample]]
        N  # negative examples per concept:  dict[str, list[sample]]
        x  # test sample (misclassified):  sample
        y  # index of correct class:  int
Return: S  # score $\in [-1, 1]$ for each concept:  dict[str, float]
1  b, t = f.layers[:L], f.layers[L:]  # Divide network $f(\cdot)$ into a bottom $b$
     (first $l$ layers) and top $t$ (remaining layers) so that $f(\cdot) = t(b(\cdot))$
2  for c in concepts: # Per concept, learn an SVM to classify bottleneck
     representations of positive and negative examples.
3    svms[c] = svm.train(b(P[c]), b(N[c]))
4    # Filter out concepts that are not learned well (i.e.  validation
     accuracies below a particular threshold).
5    if svms[c].acc < .7:
6      del svms[c]
7  # Define a concept $c$'s score as $t(b(x^*) + \delta \mathbf{v}_c)_y - t(x^*)_y$ where $t(\cdot)_y$ is the
     prob sample $x^*$ belongs to class $y$ and $\mathbf{v}_c$ is classifier boundary for $c$.
8  for c in concepts:
9    if c in svms:
10     S[c] = t(b(x) + delta*svms[c].coefs)[y] - t(b(x))[y]
11 m = max(abs(S[c]))
12 for c in S[c]:  S[c] = S[c]/m  # Normalize the scores.
```

Figure 2: **Pseudocode for CES** in Python-like syntax with explanations as comments. Learning the concepts (lines 1-6) just needs to be done once and can be carried out offline. Applying the method to each new example (lines 8-12) is very fast.

3 Methods

In this section, we detail the key steps of our method: (1) how to learn concepts that can be used to generate meaningful explanations for a wide class of test samples and (2) how to use these concepts to explain a particular misclassified sample. Pseudocode for these steps is provided in Fig. 2.

Let us define basic notation: let $f : \mathbb{R}^d \to \mathbb{R}^k$ be a deep neural network, let $\boldsymbol{x}^* \in \mathbb{R}^d$ be a test sample belonging to class $y \in \{1, \dots k\}$. We assume that that the model misclassifies $\boldsymbol{x}^*$, meaning that $\arg\max_i f(\boldsymbol{x}^*)_i \neq y$ or simply that the model's confidence in class y, $f(\boldsymbol{x}^*)_y$, is lower than desired. For readability, we will usually <u>underline</u> class names and *italicize* concepts throughout this paper.

3.1 Learning Concepts

In generating conceptual explanations, the first step is to define a concept library: a set of human-interpretable concepts $C = \{c_1, c_2, ...\}$ that occur in the dataset. For each concept, we collect positive examples, P_{c_i}, that exhibit the concept, as well as negative examples, N_{c_i}, in which the concept is absent. These concepts can be defined by the researcher, by non-ML domain experts, or even learned automatically from the data [Ghorbani et al., 2019]. Since concepts are broadly reusable within a data domain, they can also be shared among researchers.

For our experiments with natural images, we defined 160 general concepts that include (a) the presence/absence of specific objects (e.g. *mirror*, *person*), (b) settings (e.g. *street*, *snow*) (c) textures (e.g. *stripes*, *metal*), and (d) image qualities (e.g. *blurriness*, *greenness*). Many of our concepts were adapted from the BRODEN dataset of visual concepts [Fong and Vedaldi, 2018]. For each concept, we collected about 100 positive example and 100 negative examples, which we have fully released in our public repository mentioned in Section 1.

After defining a concept library, we then learn a CAV for each concept. This step only needs to be done once for each model that we want to evaluate, and the CAVs can then be used to explain any number of misclassified samples. Concretely, we pick a bottleneck layer in our model, which is the representation space in which we will learn our features. We choose the 11th layer in SqueezeNet, but for common architectures, the choice of the bottleneck layer is not too critical (see Fig. 6). Let m be the dimensionality of the bottleneck layer, and $b : \mathbb{R}^d \to \mathbb{R}^m$ be the "bottom" of the network, which maps samples from the input space to the bottleneck layer, and $t : \mathbb{R}^m \to \mathbb{R}^k$ be the "top" of the network, defined analogously.

Then, we train a regularized support vector machine to classify $\{b(\boldsymbol{x}) : \boldsymbol{x} \in P_{c_i}\}$ from $\{b(\boldsymbol{x}) : \boldsymbol{x} \in N_{c_i}\}$, the same way as in Kim et al. [2018]. For our experiments, we choose $\delta = 10,000$ though our results are not too sensitive to the this choice of hyperparameter (see Fig. 6). We denote the vector normal to the classification hyperplane boundary as $\boldsymbol{v}_{c_i}$. To measure whether the concepts are successfully learned, we keep a hold-out validation set and measure the validation accuracy, disregarding concepts with accuracies below a particular threshold (0.7 in our experiments, which left us with 156 of the 160 concepts).

3.2 Generating Conceptual Explanation Scores and Map

Once we have defined our concept bank, applying it to explain a misclassified sample can be done efficiently. We iterate over each CAV and quantify how a step perturbation in the direction of the concept in the bottleneck representation space ($\delta \cdot \boldsymbol{v}_{c_i}$) changes the prediction probability of the correct class y. Specifically, we compute the CES as the difference $t(b(\boldsymbol{x}^*) + \delta \boldsymbol{v}_{c_i})_y - t(\boldsymbol{x}^*)_y$, for each concept (see also Fig. 2). If multiple samples are misclassified, as in Fig. 3(c), then we sum over the differences $\sum_j t(b(\boldsymbol{x}_j^*) + \delta \boldsymbol{v}_{c_i})_y - t(\boldsymbol{x}_j^*)_y$. We then normalize the scores to be $\in [-1, 1]$.

A large positive score means that adding that concept to the image will increase the probability of correctly classifying the image, as will removing or reducing a concept with a large negative score. CES provides us with an assessment of which individual concepts explain a misclassified sample. To visualize the relationship and different concepts between the concepts in the concept bank, we can construct a conceptual explanation map (CEM). We reduce each CAV into a 2-dimensional embedding (e.g. using principal components analysis), and plot the CAV embeddings along with their CES scores. This can reveal clusters of concepts that explain misclassifications in a more holistic manner than individual CES (see Fig. 1(c) for an example CEM).

4 Results

In this section, we demonstrate how CES can be used to explain model limitations. First, we show that CES reveals high-level spurious correlations learned by the model. We then show analogous results for low-level image characteristics. We furthermore show that our explanations can be used to correct a model's prediction on test images in a manner similar to counterfactuals. Finally, we perform a sensitivity analysis to understand the effects of hyperparameters in our method.

4.1 CES Reveals Spurious Correlations

We start by demonstrating that CES can capture high-level spurious correlations that models may have learned. For example, consider a training dataset that consists of images of different animals in natural settings. A model trained on such a dataset may capture not only the desired correlations related to the class of animals, but spurious correlations related to other objects present in the images and the setting of the images. We can use CES to identify these spurious correlations.

To validate CES, we must know the ground-truth spurious correlations that a model has learned. To this end, we train models with intentional and known spurious correlations using the MetaDataset [Liang and Zou, 2021], a collection of labeled datasets of animals in different settings and with different objects. We construct 7 different training distributions, each consisting of 5 animal classes (<u>cat</u>, <u>dog</u>, <u>bear</u>, <u>bird</u>, <u>elephant</u>). In each model, one class is only included with a specific confounding variable (e.g. all images of <u>birds</u> are with *water*), inducing a spurious correlation in the model (images of <u>birds</u> without *water* will be misclassified), which we can probe with CES. We also train an 8th control model using random samples of animals across contexts, without intentional spurious correlations.

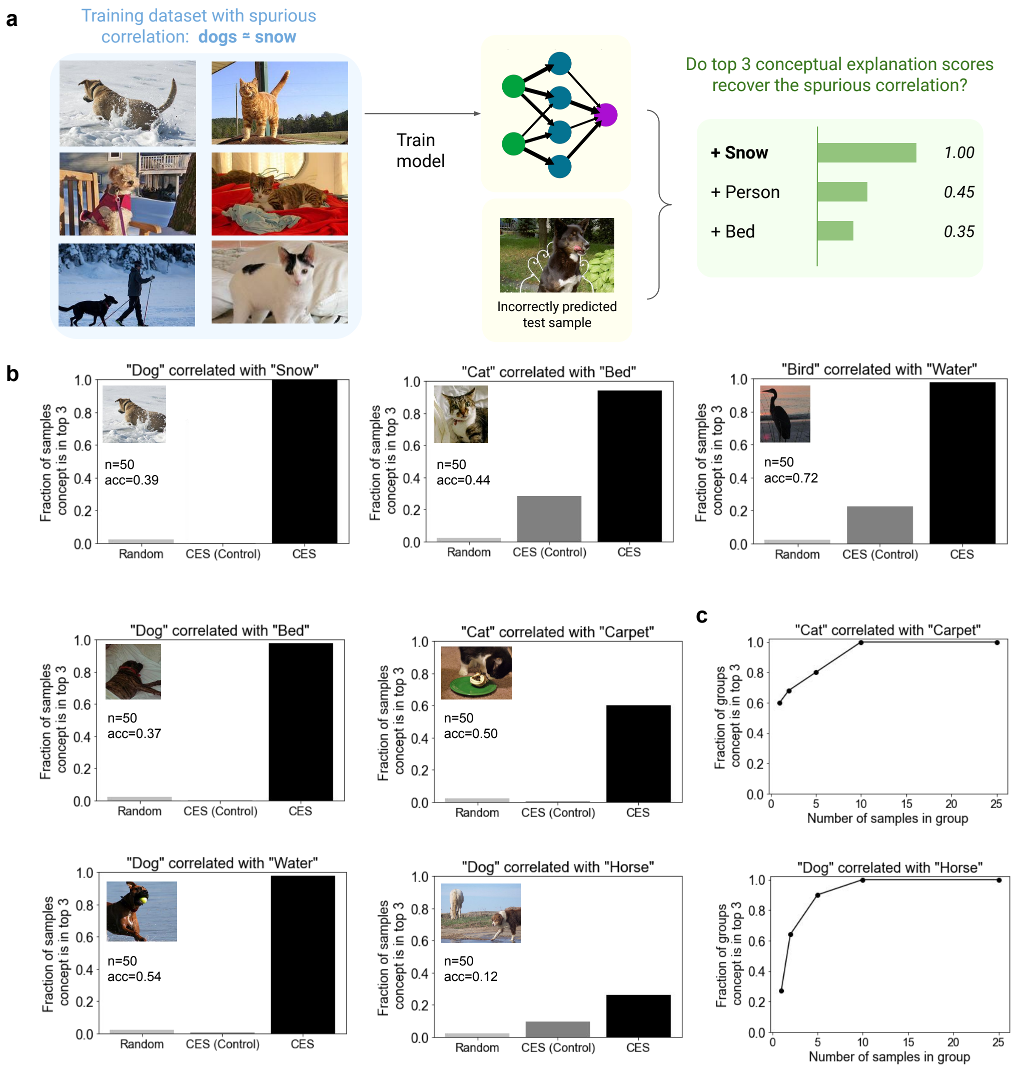

Figure 3: **Validating CES by Identifying Spurious Correlations:** **(a)** First, we train 5-class animal classification models on skewed datasets that contain spurious correlations that occur in practice. For example, one model is trained on images of dogs that are all taken with *snow*. This causes the model to associate snow with dogs and incorrectly predicts dogs without snow to not be dogs. We test whether our method can discover this spurious correlation. **(b)** We repeat this experiment with 7 different models, each that has learned a different spurious correlation, finding that in most cases the model identifies the spurious correlation in more than 90% of the misclassified test samples. We show for each model the proportion of samples in which the spurious correlation is discovered using the test images. For comparison, we also run CES using a control model without the spurious correlation, and we randomly select concepts as well. **(c)** We identify two models where the CES discovers the spurious correlation is less than 60% of test samples. Here, we can improve the performance of CES to by analyzing multiple test samples together; with just as few as 5 samples, the performance of CES exceeds 80% in both models. See Appendix A for more experimental results.

Our experimental setup is shown in Fig. 3(a). We train models with $n = 750$ images, fine-tuning a pretrained Squeezenet model. In all cases, we achieve a validation accuracy of at least 0.7. We then present the models with 50 out-of-distribution images, i.e. images of the class without the confounding variable present during training. This results in a significant drop in performance across the models (the accuracy of these models on the OOD images is shown in each panel in Fig. 3(b)).

We then use CES to recover the top 3 concepts that would explain a model's mistake on all 50 OOD images, some of whom are misclassified, while others are predicted correctly but with low confidence. We find that in the vast majority of images, CES recovers the ground-truth spurious correlation as one of the top 3 concepts. This is shown in the plots in Fig. 3(b). For example, in the model we trained with dogs confounded with *snow*, CES recovered the spurious correlation in 100% of test samples. We compare these results to performing CES using the control model, in which the same test images are presented, as well as to picking concepts randomly, both of which result in the spurious correlation being identified much less frequently.

For two of the models, CES discovers the spurious correlation less frequently (in less than 60% of test samples). These are concepts that are more heterogeneous (e.g. *carpet*) or often take up less of the image (e.g. *horse*), making them harder to learn. But even in these settings, we can improve the performance of CES to by analyzing batches of test samples together; with just as few as 5 samples, the performance of CES exceeds 80% in both models, as shown in Fig. 3(c).

4.2 CES Reveals Low-Level Artifacts

We further validate CES by showing that it can capture low-level spurious correlations and visual artifacts that models may have learned. For example, the ImagetNet dataset on which SqueezeNet was trained includes a class of green apples known as Granny Smith. These images were always colored in ImageNet [Deng et al., 2009], meaning that the model misclassifies images of these apples in grayscale.

We can use this fact to gradually transform a natural image of a Granny Smith apple, blending it with its grayscale version. At different levels of the original image vs. its transformed version, we run it through SqueezeNet to obtain a prediction, and then calculate its CES scores. The results, shown in Fig. 4(a), show that as the image is grayed, the probability of it being classified as Granny Smith decreases, while the CES for *greenness* increases. To allow comparisons across images, we do *not* normalize the CES scores. We repeat this experiment with 25 images of Granny Smith apples in Fig. 4(b). Our results show that CES is also effective at explaining a model's mistakes in terms of low-level visual artifacts, providing insights on how to change a model and training data to improve robustness. Additional examples are provided in Appendix A.2.

4.3 Concept Identified by CES are Sufficient to Correct Misclassifications

So far, we have shown that CES provides insights on how a model or training data needs to be changed to correct its prediction and improve robustness. CES can also be used, similar to counterfactual methods, to correct the prediction of a specific test image. Consider the first synthetic training dataset and model introduced in Section 4.1, which learns a spurious correlation between dog and *snow*. For a test image of a dog without snow (predicted incorrectly), CES can be generated for all 156 concepts in our concept bank.

How can we use these concepts to affect the prediction of the model? We can concatenate positive images of each concept with the test image to construct hybrid images, which include the original image but also enhance a particular concept. This follows published approaches for evaluating concepts [Kim et al., 2018]. Some examples of such images are shown in Fig. 5, along with the resulting change in the model's prediction confidence of dog. Each dot in Fig. 5(a) corresponds to a different concept concatenated to the same dog image. We show that there is a strong correlation between the CES score and the resulting change in prediction. Furthermore, we can repeat this experiment with many different test images and show the probability of the original image being predicted as dog, compared to the hybrid image. We show the results with the top concept, *snow*, showing that the hybrid images have consistently higher probability. These experiments provide validation that CES is scoring concepts effectively and in a way can be used to successfully change a model's prediction in the desired direction.

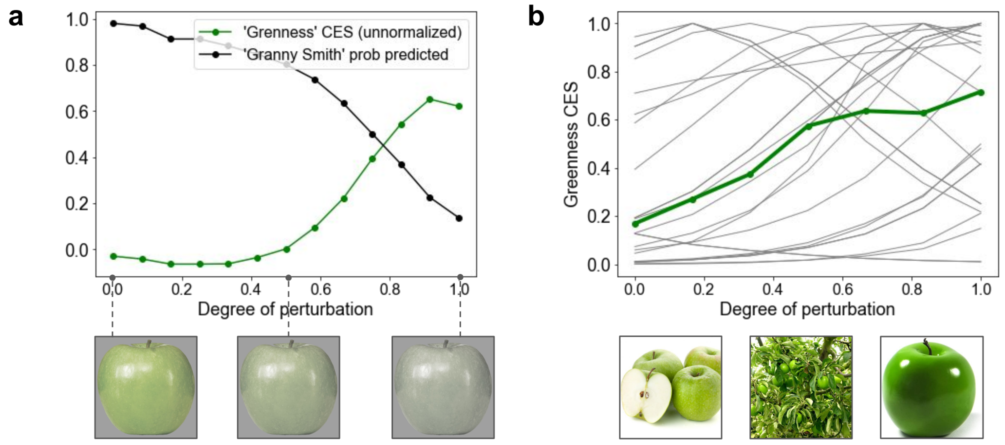

Figure 4: **Validating CES Through Low-Level Image Perturbations:** **(a)** Here, we take an arbitrary test sample of a Granny Smith apple that is originally correctly classified and perturb the image by turning it gray until it is eventually misclassified (as mortar). We compute the *unnormalized* CES scores at each perturbed input image and observe that the score for *greenness* increases, corresponding to the degree to which we remove the green color from the image. **(b)** We repeat this for 25 different images of Granny Smith apples (some of which are shown under the plot) and find that the same trends generally hold true (each image is a gray line). Although a few images do not follow this trend, the mean CES score (bolded green line) does.

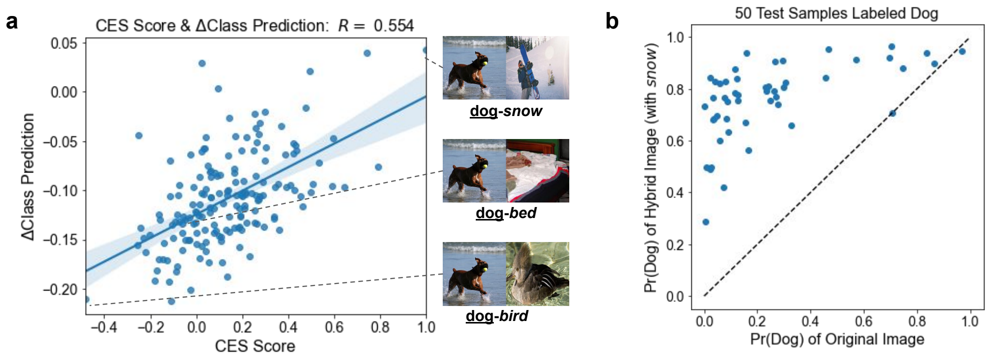

Figure 5: **Changing Class Predictions Through Hybrid Images:** Here, we explore how we can change the predictions of a model (trained with a spurious correlation between dogs and *snow*) on misclassified test samples. **(a)** We take one misclassified image of a dog (predicted as dog with probability 0.23) and construct hybrid images of the dog concatenated with images of different concepts. We notice that the likelihood of the model classifying the resulting image as dog is associated with the CES score for that concept ($R = 0.554$), providing further validation to CES. Each dot here represents one concept, and the best fit line in shown in solid blue. See Appendix A.3 for experiments with multiple test images. **(b)** We use this idea to pick the top concept (*snow*) and create hybrid images with 50 test samples of dogs, many of whom are misclassified as not dogs. Each dot in this panel corresponds to a different dog image concatenated with a snow image. We notice that the probability of the hybrid image being classified as dog is much higher than the original probability, across a range of original probabilities. This validates that the top concept (*snow*) identified by CES is sufficient to correct the model's initial misclassifications.

4.4 Sensitivity Analysis

Finally, we conduct experiments to investigate the two hyperparameters in our algorithm for CES described in Fig. 2: the choice of the bottleneck layer L, and the step size δ. First, we show that the choice of the layer does not seem to have a significant effect in the accuracy of the linear models used to learn concepts. The validation accuracies of five represntative concepts are shown in Fig. 6(a). Furthermore, when these different layers are used to compute CES, the results are highly correlated as shown in Fig. 6(b). We also change the step size across 5 orders of magnitude, and find that the CES scores do not change significantly. This suggests that our method is fairly robust to changes in the two hyperparameters.

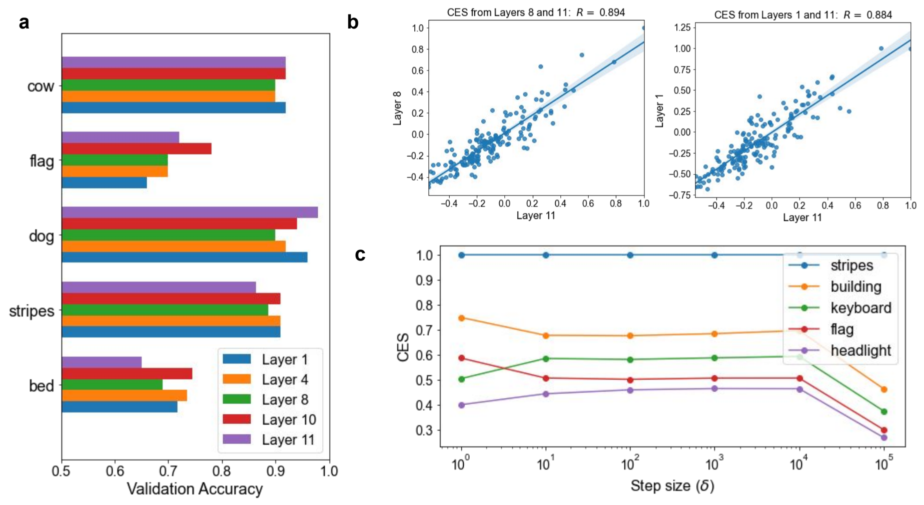

Figure 6: **Sensitivity Analysis for Hyperparameters:** Our algorithm for computing CES consists of two hyperparameters, but we show that the method is robust to changes in both. **(a)** Classifiers are trained on 5 representative concepts in different bottleneck layers of Squeezenet. We find that the validation accuracies do not vary significantly from one layer to another. They do vary from one concept to another. **(b)** Furthermore, we compute CES scores for an arbitrary but representative test image (namely, the top zebra image in Fig. 1), finding that they are highly correlated when bottleneck layers close to one another are used (left, $R = 0.894$) or even when layers far apart from one another used (right, $R = 0.884$). Each blue dot is one concept and the best-fit line is solid blue. **(c)** We also vary the step size δ across 5 orders of magnitude and find that it does not significantly effect the CES scores, particularly for top concepts shown here. We also conduct experiments with other network architectures and find similar results, shown in Appendix A.4.

5 Conclusion and Future Work

The method that we have presented, CES, provides a meaningful step towards explaining why machine learning models make mistakes on test samples. Our method is fast, can be readily applied to any deep network without retraining, and provides explanations in human-interpretable terms. We have validated the method quantitatively, showing that it can be used to discover spurious correlations and other artifacts learned by models.

There are several promising areas where CES can be extended. For example, our method scores concepts individually, while it may be useful to develop methods that can better capture interrelationships between concepts. Furthermore, while we have focused in this paper on image classification tasks, CES can be applied to other data modalities such text, audio, and video data, as well as other tasks such as regression and segmentation. Even within image classification, we can apply CES to more specialized domains (e.g. pathology images) by developing or discovering concepts relevant to users in those fields. Finally, we seek to do user studies to understand how human subjects respond to explanations with CES and how it drives improvements in model debiasing and robustness.

References

A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019.

J. Adebayo and L. Kagal. Iterative orthogonal feature projection for diagnosing bias in black-box models. *arXiv preprint arXiv:1611.04967*, 2016.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *arXiv preprint arXiv:1802.07623*, 2018.

R. Fong and A. Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8730–8738, 2018.

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

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

D. Kiela, M. Bartolo, Y. Nie, D. Kaushik, A. Geiger, Z. Wu, B. Vidgen, G. Prasad, A. Singh, P. Ringshia, et al. Dynabench: Rethinking benchmarking in nlp. *arXiv preprint arXiv:2104.14337*, 2021.

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*, pages 2668–2677. PMLR, 2018.

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

W. Liang and J. Zou. Metadataset: A dataset of datasets for evaluating distribution shifts and training conflicts. *arXiv preprint*, 2021.

Z. Lipton, Y.-X. Wang, and A. Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

D. Mahajan, C. Tan, and A. Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.

B. Nestor, M. B. McDermott, W. Boag, G. Berner, T. Naumann, M. C. Hughes, A. Goldenberg, and M. Ghassemi. Feature robustness in non-stationary health records: caveats to deployable model performance in common clinical machine learning tasks. In *Machine Learning for Healthcare Conference*, pages 381–405. PMLR, 2019.

A. Raghunathan, J. Steinhardt, and P. Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *arXiv preprint arXiv:1811.01057*, 2018.

A. Subbaswamy, P. Schulam, and S. Saria. Preventing failures due to dataset shift: Learning predictive models that transport. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3118–3127. PMLR, 2019.

S. Verma, J. Dickerson, and K. Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2020. doi: 10.1109/TVCG.2019.2934619.

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]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (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? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (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]
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? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]

A Additional Experimental Results

The appendix contains additional experimental results and figures that further support the findings in the main paper.

A.1 Additional Examples Validating CES by Identifying Spurious Correlations

In Fig. 3, we showed 7 case studies in which CES was successfully able to identify spurious correlations in the dataset. Here, we take 3 of those examples. In each case, the model has learned a spurious correlation between dog and a different concept. We display the most common concepts that appear as one of the top 3 concepts recovered by CES. Examining these concepts sheds further insights into the spurious correlations learned by the model.

A.2 Additional Examples Validating CES through Low-Level Image Perturbations

In Fig. 4, we showed an example in which CES was successfully able to identify why images that are being perturbed through low-level image transformations is being misclassified. Here, we show another pair of examples, with a different concept: *redness* (Fig. 10).

A.3 Multiple Hybrid Images

In Fig. 5, we showed how CES scores can be used to change image class predictions in a particular direction. In Fig. 5(a) was constructed using a single test image. Here, in Fig. 9, we show the results when averaged over 3 different test images, for each concept.

A.4 Multiple Architectures

In Fig. 6(a) and (b), we showed that CES scores are quite robust to choice of layer within a network. Here, we show the results for different networks altogether. We test 3 different architectures, all trained on ImageNet. In each case, we use the penultimate layer of the network as the bottleneck layer, and find that the overall validation accuracies and even the CES scores are quite highly correlated. Results are shown in Fig. 10.

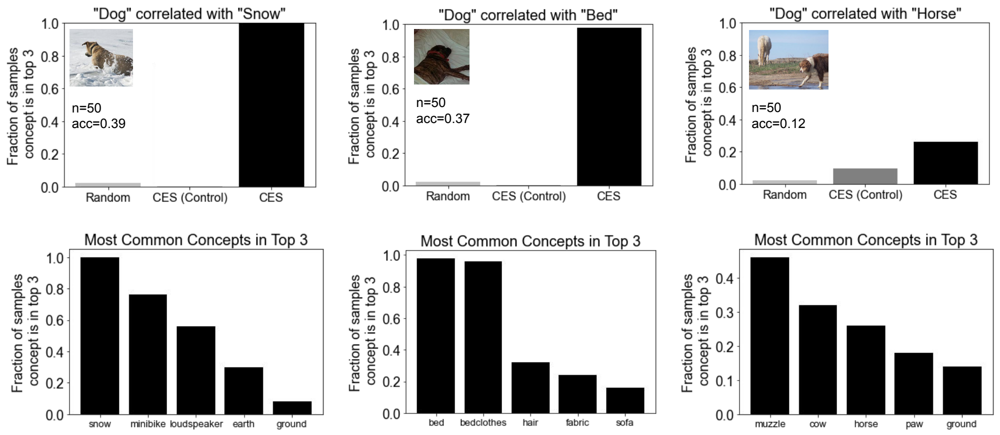

Figure 7: As in Fig. 3, we train 5-class animal classification models on skewed datasets that contain spurious correlations that occur in practice, and determine whether CES is able to successfully uncover the correlated concept (top). Here, we take 3 of those examples and examine the other concepts that are returned by CES (bottom). Examining these concepts sheds further insights into the spurious correlations learned by the model.

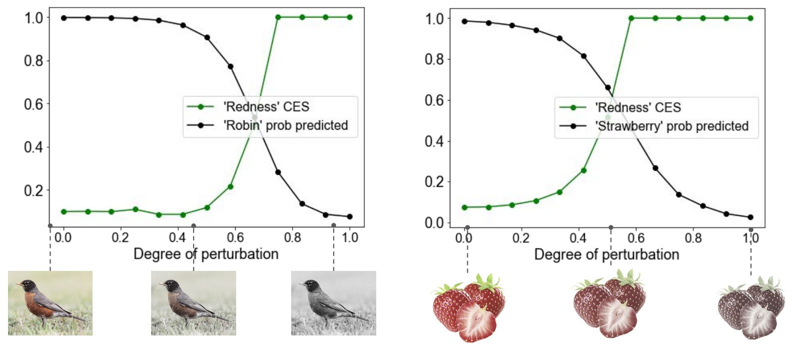

Figure 8: In an analogous manner to Fig. 4, we take images that were originally correctly classified as robin and strawberry, and perturb them by removing red colors until they are misclassified by the model (x-axis). The CES scores correctly identifies the concept of *redness* as the most important concept for correcting the model's mistakes.

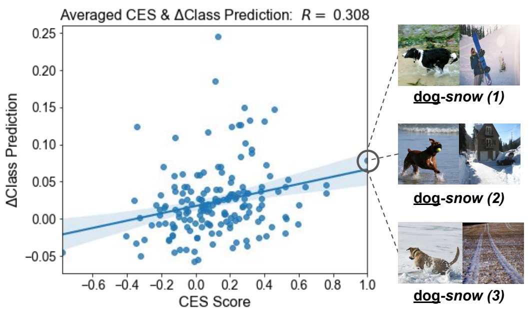

Figure 9: Here, we explore how we can change the predictions of a model (trained with a spurious correlation between dogs and *snow*) on misclassified test samples. We take three misclassified image of a dog and construct hybrid images of the dogs concatenated with images of different concepts. Each data point is the average CES score for the concepts averaged over the three images, plotted against the average change in class prediction when we construct three hybrid images. For example, the x-value of dot on the far right corresponds to three misclassified dog images, after averaging their CES scores for *snow*. Its y-value corresponds to the difference in class prediction score for class dog between the original images of dogs and those images concatenated with three images representing the concept *snow*. We notice that the increase in the likelihood of the model classifying the resulting the resulting hybrid image as dog is associated with the CES score for that concept ($R = 0.308$)

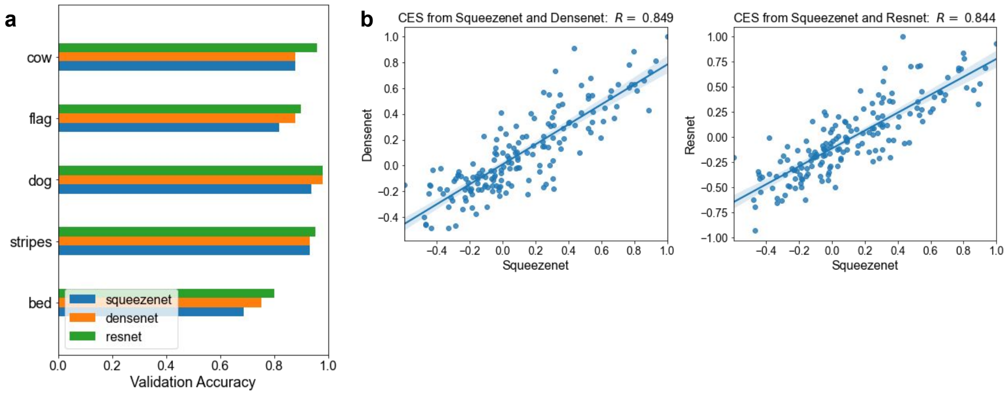

Figure 10: (a) Validation accuracies of various concepts in different architectures for some representative concepts. We find that the validation accuracies do not vary significantly from one layer to another. They do vary from one concept to another. (b) The CES score for Squeezenet vs. Densenet (left), and the CES score for Squeeznet vs. Resnet (right). Each dot corresponds to one concept for an arbitrary but representative misclassified test sample (namely, the top zebra image in Fig. 1).