Meaningfully Explaining a Model's Mistakes

Abubakar Abid [1] James Zou [2]

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 our 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. The code for CES is publicly available and can easily be applied to 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?* Answering this question correctly affects the *usability* of a model and can help make the model more *robust*.

Example: 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 also make it more robust (Abid et al., 2019; Kiela et al., 2021), and 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*).

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*. Here, we review these methods and discuss differences between them and our proposed approach.

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

*Equal contribution [1]Department of Electrical Engineering, Stanford University, Stanford, United States [2]Department of Biomedical Data Science, Stanford University, Stanford, United States. Correspondence to: James Zou <jamesz@stanford.edu>.

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 compute CES, we use a similar approach, perturbing input data to change its predictions, but for a complementary goal: to understand the limitations 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.

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, our approach allows a more contextual understanding of a model's behavior and mistakes.

3. Methods

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.

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.

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 for experiments in this paper, but for common architectures, the choice of the bottleneck layer is not too critical. Let m be the dimension 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, and $t : \mathbb{R}^m \to \mathbb{R}^k$ be the "top" of the network, defined analogously.

Then, we train a 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). We denote the vector normal to the classification hyperplane boundary as $\boldsymbol{v}_{c_i}$. To measure whether concepts are successfully learned, we keep a hold-out validation set and measure the validation accuracy, disregarding concepts with accuracies below a threshold (0.7 in our experiments, which left us with 156 of the 160 concepts).

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. For our experiments, we choose $\delta = 10,000$ though our results are not too sensitive to the this choice of hyperparameter. 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. If multiple samples are misclassified, as in Fig. 1(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.

4. Results

In this section, we demonstrate how CES can be used to explain model limitations. We start by demonstrating that CES can capture high-level spurious correlations that models may

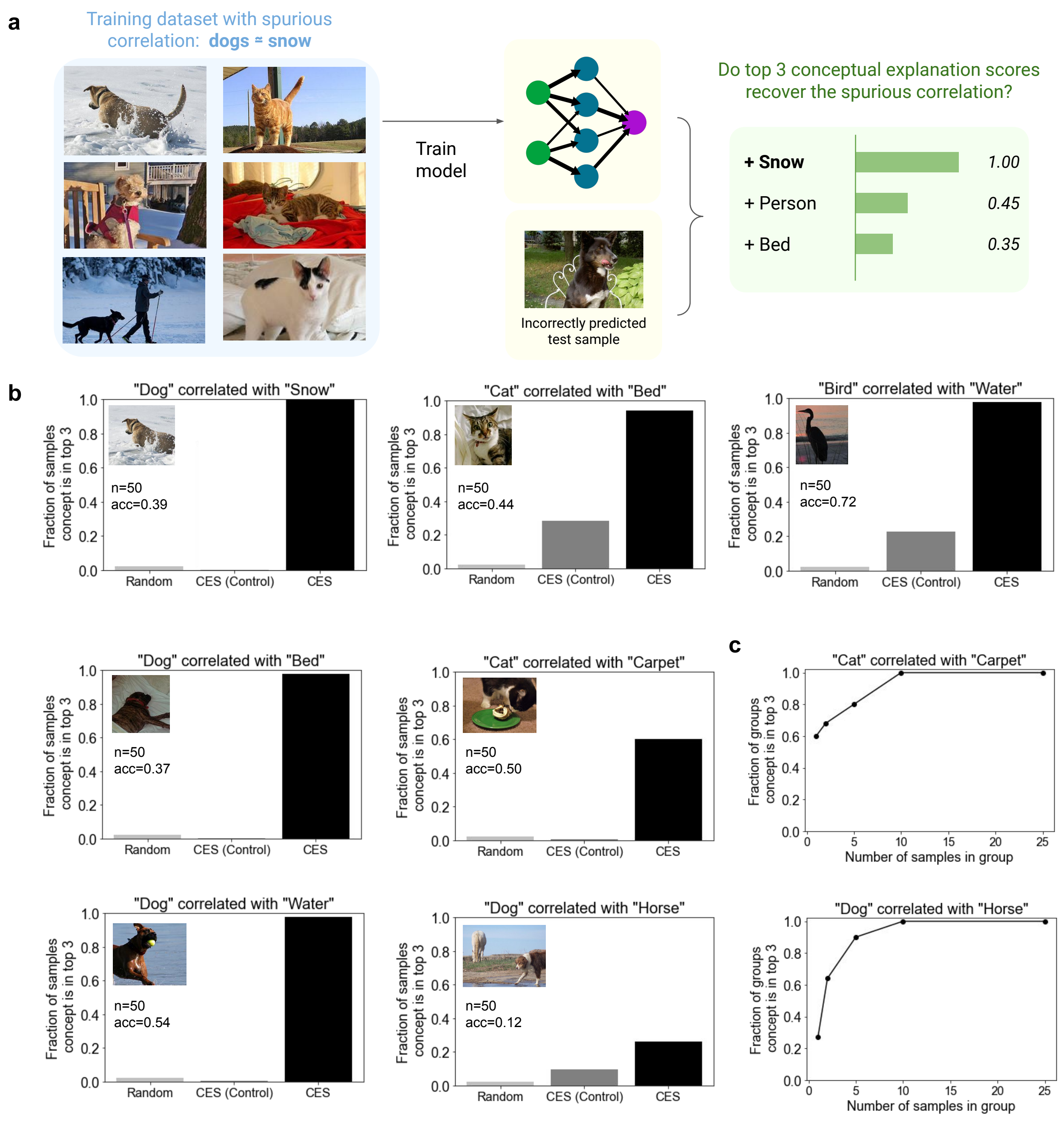

Figure 1. **Validating CES by Identifying Spurious Correlations:** **(a)** First, we train 5-class animal classification models on skewed datasets that contain spurious correlations that occur in practice. For example, one model is trained on images of dogs that are all taken with *snow*. This causes 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.

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 (cat, dog, bear, bird, elephant). In each model, one class is only included with a specific confounding variable (e.g. all images of birds are with *water*), inducing a spurious correlation in the model (images of birds 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.

Our experimental setup is shown in Fig. 1(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 (OOD) 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. 1(b)).

We then use CES to recover the top 3 concepts that would explain a model's mistake on each of 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. 1(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. 1(c).

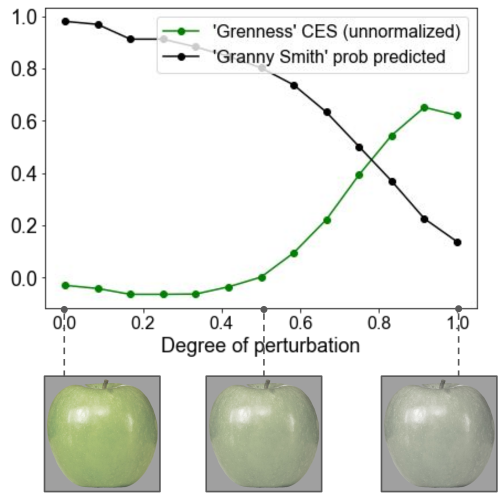

Figure 2. **Validating CES Through Low-Level Image Perturbations:** Here, we take an arbitrary test sample of a Granny Smith apple that is originally correctly classified and perturb the image by turning it gray until it is eventually misclassified (in this case, as mortar). We compute the *unnormalized* 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.

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. 2(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. 2(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.

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. 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. 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.

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.

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.

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.