Quantitatively Evaluating the Validity of Contrastive Generators for Recourse

Ian Howell[1], Eleanor Quint[1], Hongfeng Yu[1,2]
[1]*School of Computing, University of Nebraska-Lincoln, Lincoln, NE, USA*
[2]*Holland Computing Center, University of Nebraska-Lincoln, Lincoln, NE, USA*

***Abstract*—Explanations of classifiers can provide recourse to those impacted, i.e., the ability to facilitate those impacted to understand and potentially change the classification. Contrastive explanations provide this recourse by producing an alternative input close to the original such that the label is changed to the desired, implying actions in the difference between the original and contrastive inputs. However, quantitatively evaluating contrastive explanations remains a challenging task. In particular, some state-of-the-art contrastive explanation algorithms for decisions made by deep neural networks can produce inputs that are out-of-distribution or adversarial and are thus either infeasible for a user to achieve or do not change the label according to the underlying data distribution, respectively. Past work has termed contrastive examples *valid* if they cross the decision boundary of the classifier. However, this definition does not encompass these failure modes and thus is not suitable to evaluate contrastive methods when used for recourse. In this paper, we define a new type of validity, called *distributional validity*, that checks for these failure modes. We experiment with the distributional validity of state-of-the-art contrastive explanation methods and find that the best contrastive method depends on the architecture of the classification model.**

***Index Terms*—explainable AI, classification, recourse, contrastive explanation, validity**

I. Introduction

Machine-learning based classifiers have demonstrated performance on par with or close to humans in many settings and are increasingly used to influence critical decisions in diverse fields, such as medical diagnosis, aiding sentence recommendations, and so on [1], [2]. However, these algorithms are often perceived as black-box systems due to their intricate operations and structures, such as deep learning models with numerous interconnected layers and parameters. Substantial efforts have focused on enhancing the explainability of these systems [3], thereby empowering users with *recourse*, i.e., the ability to understand classification decisions better and make necessary adjustments to achieve desirable results from classifiers [4]. Among these endeavors, *contrastive* or *counterfactual explanations* have proven effective in helping users understand model behaviors and providing actionable recommendations for recourse [5], [6]. In general, contrastive explanations answer questions of the form "Why did the classifier label X as P instead of as Q?" by providing an alternative input X', called a *contrastive example*, that is close to the original such that the classifier predicts Q instead of P [7].

While many contrastive explanation generation methods have been developed, evaluating their resulting contrastive examples is non-trivial, where evaluation metrics proposed by researchers include validity, actionability, proximity, diversity, and so on. In particular, among these metrics, quantitatively evaluating the *validity* of contrastive explanations remains an open problem [5], [8], which can be illustrated in Figure 1 where points B, C, D, E, and F are possible outputs of a contrastive example algorithm for a given input point A.

First, a contrastive explanation generator may fail to move the data across the model's decision boundary and *approximately* the boundary of the conditional, like points B and E. Second, the generated data may fall outside the data distribution, such as point C, which would mean that it is not plausible for users to change their situation to that represented by C. Third, the explanation method may produce alternatives that fall within the data distribution, and thus are feasible for users to move to, but on which the model and data distribution disagree, such as point D. Finally, while point E is both in-distribution and has crossed the boundary of the underlying labeling function, it causes disagreement between the classifier and the true labeling function, an undesirable behavior of explanation as it relies on a failure of the model. Point F is the only of those depicted that is feasible and crosses the decision boundaries of both the model and the true labeling function. Past work has called points C, D and F *valid* contrastive examples [5], [8]; however, C and D give explanations that do not match the underlying conditional distribution.

In this paper, we address the validity of contrastive explanations by focusing on the cases where contrastive examples lie outside the data distribution (point C) or cause classifier-process disagreement (points D and E). These failure modes are especially important to contrastive explanations because some methods of contrastive generation rely on optimizing the input with respect to a desired classifier output [9]–[11], a technique that closely aligns with those used to perform *adversarial attacks*, i.e. perturbations that change the classifier's predicted label while still looking like natural data [12]–[15]. However, these failure modes are not considered by previous work when determining whether contrastive examples are valid [5], [8].

To tackle this issue, we note that these failure modes can be approximately detected using out-of-distribution (OOD) and adversarial detection methods, which we elaborate on in Section III. Based on our observation, we propose to define *distributionally valid* (DV) contrastive examples to fall within the data distribution and cross the classification boundaries

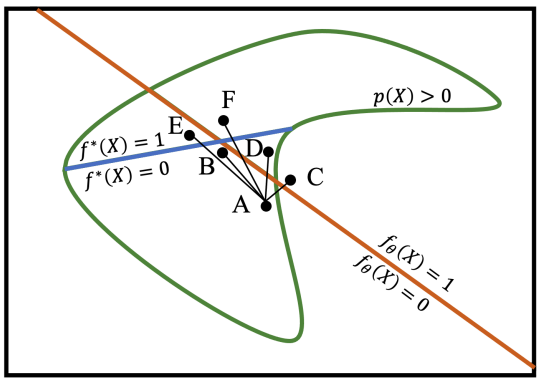

Fig. 1. Examples of different possible results from generating contrastive explanations. The green line encapsulates a data manifold $p(\mathbf{X}) > 0$, with the blue representing the boundary of the true labeling function f^*. Note, the blue line only extends where the data has support, i.e., $p(\mathbf{X}) > 0$. The orange line divides the entire set X by the decision boundary of a model, f_θ. The contrastive question is "Why is $f_\theta(A) = 0$ and not $f_\theta(A) = 1$?", with points B, C, D, E, and F being possible outputs of a contrastive example algorithm. Points B, C, D, and E exhibit undesirable behavior. Point B is close to the boundary of both the model and the underlying labeling function; however, it does not actually cross either. Point C causes the model to output $f_\theta(C) = 1$; however, it is out of the data distribution altogether. Points D and E are both in-distribution; however, they cause disagreement between the model f_θ and the true labeling function f^*. Finally, Point F crosses decision boundaries set by both the model and the underlying labeling function, making it the best contrastive example of those discussed. While C and D are termed *valid* in the existing work, they are not useful for recourse as moving from A to either C or D will not change the label in the underlying labeling function.

defined by both the model and conditional distribution. We use this definition to analyze contrastive generation methods using OOD and adversarial detection methods to approximate the proportion of constructed examples that are, in fact, DV. The contributions of this paper include:

- Defining the DV property of contrastive examples and a method that approximates their proportion in the outputs of a contrastive method.
- Conducting experiments that demonstrate estimating the proportion of samples that are DV with a family of OOD and adversarial detectors, which we claim is necessary for contrastive examples used for recourse but cannot be adequately characterized using existing validity property.

Our findings show the superiority of DV over prior research in quantitatively evaluating the validity of contrastive generators. We utilize a range of DV detectors and demonstrate their effectiveness in tackeling failure modes of contrastive examples that cannot be fully captured by existing solutions. Our newly devised distributional validity opens more opportunities for comprehending contrastive explanations across various application domains.

II. Background and Related Work

We review relevant methods to generate and evaluate contrastive examples, as well as methods of adversarial attacks, and detecting OOD and adversarial data.

A. Contrastive Example Generation Methods

Contrastive or counterfactual example generators find changes to input data to activate a different set of labels or neurons. The change to the input should make minimal, but perceptually sufficient, changes to the label-relevant semantics. Adversarial attacks have the same objective, but remove the "perceptually sufficient" requirement to try to avoid detection.

The activation maximization (AM) framework aims to find inputs that produce specific results from a model [16]. Nguyen *et al.* [17] propose to use a deep generative network to help guide the optimization, as the generative network helps keep data samples on the data manifold. Given a target class y_{tar}, the authors generate a latent code $\hat{\mathbf{z}}$ of the prototype with:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}\in\mathbb{R}^d}{\operatorname{argmax}} \ \log p[y_{tar}|G_\phi(\mathbf{z})] - \lambda||\mathbf{z}||, \tag{1}$$

where G_ϕ is a deep generative network, and λ is a regularization term on the latent code of the generator, $\mathbf{z}$. To generate data that exhibits predictions by the classifier closer to that of predictions of data within the training distribution, Joshi *et al.* [9] propose xGEMs, which uses the loss function with which the classifier was trained. Further, they consider the perceptual distance between the original input $\mathbf{x}_0$ and the contrastive example $G_\phi(\mathbf{z})$. This results in the optimization:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}\in\mathbb{R}^d}{\operatorname{argmin}} \ \mathcal{L}(\mathbf{x}_0, G_\phi(\mathbf{z})) + \lambda l(f_\theta(G_\phi(\mathbf{z})), y_{tar}), \tag{2}$$

where $\mathcal{L}$ is a distance in input space, and l is the original loss function of the classifier. For the above methods, the prototype and contrastive example are recovered by $\hat{\mathbf{x}} = G_\phi(\hat{\mathbf{z}})$. To ensure that the boundary between the original class and the target class is in focus, Feghahati *et al.* [10] propose CDeepEx, which instead solves the constrained optimization problem:

$$\begin{aligned} \hat{\mathbf{z}} = \ & \underset{\mathbf{z}}{\operatorname{argmin}} \ ||\mathbf{z} - \mathbf{z}_0||_2^2 \\ \text{s.t.} \ & \log p(y|I_{\mathbf{z},\mathbf{z}_0}) - \log p(y_{tar}|I_{\mathbf{z},\mathbf{z}_0}) = 0 \\ & \log p(y|I_{\mathbf{z},\mathbf{z}_0}) - \log p(y'|I_{\mathbf{z},\mathbf{z}_0}) \geq \varepsilon \\ & \log p(y_{tar}|I_{\mathbf{z},\mathbf{z}_0}) - \log p(y'|I_{\mathbf{z},\mathbf{z}_0}) \geq \varepsilon \\ & \forall y' \neq y, y' \neq y_{tar}, \end{aligned} \tag{3}$$

where $\mathbf{z}_0$ is the latent code corresponding to the original image via $\mathbf{z}_0 = \operatorname{argmin}_z ||G_\phi(\mathbf{z}) - \mathbf{x}_0||_2$, $I_{\mathbf{z},\mathbf{z}_0} = G_\phi(\mathbf{z}) + \Delta_{\mathbf{z}_0}$, and $\Delta_{\mathbf{z}_0} = G(\mathbf{z}_0) - \mathbf{x}_0$. However, as the classifier is trained such that one class is selected to have a high probability over the others, inputs that require equal log probabilities between two classes may be prone to being out of the training distribution. Poyiadzi *et al.* [6] address the feasibility and actionability of contrastive examples by grounding their definitions in the density of the training distribution. They find contrastive samples in training sets via pathfinding, sticking to high-probability data. However, their method does not consider whether these high-probability points are adversarial with respect to the classifier. Our method explicitly includes these noted failure cases in calculating DV contrastive examples.

B. Evaluating Contrastive Examples

Mothilal *et al.* [8] test the *target class validity* (TCV) of a contrastive example, i.e., whether the classifier predicts the label of the example to be the target class. This method classifies points C and D in Figure 1 as being valid, even though they do not cross the boundary of the conditional or even fall outside the data distribution. Looveren *et al.* [11] define an interpretable contrastive example as one that lies close to the model's training distribution and introduce two metrics. The first, IM1, approximates whether an example is closer to the distribution of data with the contrastive label or to the distribution of data with the original label. The second, IM2, approximates if a contrastive example is described equally well by the distribution of the contrastive class and the entire data distribution, signaling a more interpretable example. Both of these methods work toward determining if a contrastive example lies within the same distribution as training data with the contrastive label. While these approaches are able to measure certain aspects of contrastive explanations, they do not account for adversarial perturbations that can arise from contrastive optimization, like points D and E in Figure 1.

Laugel *et al.* [18] introduce methods to calculate the proximity, connectedness, and stability (or robustness) of contrastive explanations. The most similar metric to our own is the stability of a contrastive example, where a contrastive example is stable if data that is close to it exhibit similar explanations. However, this is complementary to our method, as a contrastive example may be stably adversarial.

C. Adversarial Attacks

Adversarial attacks apply *imperceptible* perturbations to data that cause the prediction output by a model to disagree with a conditional distribution [15]. Many works keep perturbations imperceptible by minimizing perturbation magnitudes, e.g., in terms of a pixel-norm [13], [14], [19] or Just Noticeable Differences in image space [20]. The Fast Gradient Sign method (FGSM) [19] takes a single step on the loss gradient of the classifier that maximizes the loss of a network f_θ, bounding the L_∞ norm of the change by a small ϵ:

$$\hat{\mathbf{x}} = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)), \tag{4}$$

where $J(\cdot)$ is the loss function used to train the classifier. The Basic Iterative Method (BIM) [21] extends this idea by taking multiple, smaller optimization steps on the same gradient. These methods may be used to target a specific class y' by using the targeted class in the loss function. The Carlini & Wagner L_2 (CWL2) approach [14] is a more powerful technique that minimizes:

$$\begin{aligned} \hat{\mathbf{w}} = \quad & \underset{\mathbf{w}}{\text{argmin}} \, || \frac{1}{2}(\tanh(\mathbf{w}) + 1) - \mathbf{x}_0 ||_2^2 \\ & + c \cdot f\left(\frac{1}{2}(\tanh(\mathbf{w}) + 1)\right), \end{aligned} \tag{5}$$

where $f(\mathbf{x}) = \max(\max\{Z(\mathbf{x})_i : i \neq t\} - Z(\mathbf{x})_t, -\kappa)$, Z is the logits of the classifier, t is the target label, and the adversarial example $\hat{\mathbf{x}}$ is recovered by $\hat{\mathbf{x}} = 1/2 \cdot (\tanh(\hat{\mathbf{w}}) + 1)$. Given $\kappa = 0$, a default provided by the authors, the logits of the target and original classes are pushed to be equal in the optimization. In our work, we train adversarial detectors to recognize these three attacks to help us determine whether contrastive examples are adversarial.

D. Adversarial Detection

Many methods have been designed to deal with the above adversarial attacks, either training the model using adversarial augmentation [22] or by detecting adversarial examples post-hoc, i.e., after model training. Techniques that fall into the latter category are of particular interest to this work, as they function with any model that provides a gradient and do not require retraining the classifier, which incurs additional cost. Feinman *et al.* [23] propose to estimate the density of the training distribution's representation in the last hidden layer of a convolutional neural network. This estimate is then used to determine what inputs lie far from the data manifold. Ma *et al.* [24] propose the local intrinsic dimensionality (LID) metric to estimate the input distance distribution, which differs for adversarial examples that lie close to but not on the data manifold. Lee *et al.* [25] calculate the Mahalanobis distance of activations at each block of a network from those collected from the training set. The authors then train a logistic regression model that is able to detect adversarial examples. They also show that their method can accurately detect OOD samples. In our work, we train LID and Mahalanobis-based detectors to detect adversarial contrastive examples.

E. Out of Distribution Detection

Liang *et al.* [26] propose **O**ut-of-**DI**stribution detector for **N**eural networks (ODIN), a multistep OOD detector. ODIN augments the input by taking an optimization step to minimize the loss of the classifier with respect to the predicted class of the classifier. It then calculates a maximum, temperature-scaled softmax value of the classifier. A logistic regression model is learned on this softmax score to detect OOD samples. Ren *et al.* [27] propose to measure the log likelihood ratio (LLR) between a standard density model and a background density model that has been trained on corrupted data. If the difference in likelihood is low, then the background statistics play a large role in the likelihood and therefore the example is labeled as out-of-distribution. Additionally, as mentioned above, the Mahalanobis-based detection method of Lee *et al.* [25] works well in detecting OOD samples. Our method uses these techniques to determine OOD contrastive examples.

III. Methods

In this section, we derive and define distributionally valid contrastive examples and how to automatically detect contrastive examples that exhibit this property.

Let $p(\mathbf{X}, Y)$ be the joint distribution of the data with a true labeling function f^* that corresponds to the point distribution $p(Y|\mathbf{X})$. Let $(\mathbf{x}, y) \sim p(\mathbf{X}, Y)$ be an input with its associated label, such that the classifier $f_\theta(\mathbf{x})$ agrees with the true

classifier $f^*(\mathbf{x})$, i.e., $f_\theta(\mathbf{x}) = f^*(\mathbf{x})$, an assumption made for the setting of recourse rather than model debugging. Suppose a user wishes to understand why $\mathbf{x}$ was not classified as another label, y'. They apply a contrastive explanation method h to generate the example using f_θ as an approximation of f^*, $\mathbf{x}' = h(\mathbf{x}, y')$. Previous work defines $\mathbf{x}'$ to be a valid contrastive example if $f_\theta(\mathbf{x}') = y'$ [5], [8]. This definition considers only the classification boundary learned by the classifier f_θ and not the data manifold $p(\mathbf{X})$, shown by the green line in Figure 1, nor the disagreement between f_θ and f^*, exemplified by points D and E.

To better account for the data manifold and conditional data distribution, we define a *distributionally valid* (DV) contrastive example to be a valid contrastive example such that the true labeling function f^* agrees with the classifier for the contrastive label, i.e., $f^*(\mathbf{x}') = f_\theta(\mathbf{x}')$:

$$DV(\mathbf{x}', y') = \mathbb{1}(f_\theta(\mathbf{x}') = y' \wedge f^*(\mathbf{x}') = f_\theta(\mathbf{x}')). \quad (6)$$

While $f_\theta(\mathbf{x}') = y'$ is easily testable, it is intractable to determine the agreement of the classifier with the true labeling function. To address this challenge, we partition model errors based on whether data has a true label, i.e., whether the marginal $p(X)$ has support. In the case that it does not, i.e., $p(X) = 0$, then x is out-of-distribution like point C, a case that can be approximately detected by various methods discussed in Section II-E. In the case that $p(X) > 0$, then the classifier has simply misclassified the data, like points D and E. We argue that for a well-trained classifier, this data can be approximately detected using an adversarial detector as the data must be close to in-distribution data in the input space. This is supported by empirical evidence that adversarial detectors trained to detect one type of attack are in-fact good detectors of other attacks, i.e., other data near the boundary that the classifier misclassified ([25], Section IV-B).

Therefore, we approximate agreement between the true labeling function f^* and the classifier f_θ using the combination of an OOD detector, $p_\phi(\mathbf{x}) > 0$, and an adversarial detector $Adv_\psi(\mathbf{x})$ that returns 1 if $\mathbf{x}$ is adversarial with respect to f_θ and 0 otherwise:

$$\begin{aligned} DV(\mathbf{x}', y') \approx & \mathbb{1}(f_\theta(\mathbf{x}') \\ & = y' \wedge p_\phi(\mathbf{x}') > 0 \wedge Adv_\psi(\mathbf{x}') = 0). \end{aligned} \quad (7)$$

Finally, for a dataset $D \sim p(\mathbf{X})$, we define the distributional validity of the contrastive generator h to be the proportion of generated examples that are DV:

$$DV(h) = \frac{\sum_{\mathbf{x} \in D} \sum_{y \in Y s.t. y \neq f^*(\mathbf{x})} DV(h(\mathbf{x}, y'), y'; \theta)}{|D|(|Y| - 1)}. \quad (8)$$

IV. Experimental Results

We have conducted detailed experiments to systematically evaluate the distributional validity developed in our approach. Our experiments incorporate different architectures, hyperparameters, and training configurations (Section IV-A). We first assess and select suitable OOD and adversarial detectors according to their abilities to differentiate in-distribution data from OOD or adversarial data, and the selected detectors allow us to meaningfully approximate the DV of contrastive examples (Section IV-B). We also evaluate the ability of these detectors to detect out-of-distribution and adversarial data (Section IV-C). Finally, we use the best-performing detection methods we found to evaluate the validity of contrastive example generators with the existing validity metrics and our devised DV and compare the effectiveness of these metrics (Section IV-D).

A. Architectures, Hyperparameters, and Training

1) Contrastive Example Generation Methods: In these experiments, we evaluate contrastive examples generated by Activation Maximization (AM), xGEMs, and CDeepEx. To generate contrastive examples using the AM approach of Nguyen *et al.* [17], we modify Equation 1 to penalize the change in the latent space variable z_0 using an L_2 norm:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmax}} \ \log p[w_{y'} | G_\phi(\mathbf{z})] - \lambda ||\mathbf{z} - \mathbf{z}_0||_2, \quad (9)$$

where G_ϕ is a generative model, $\mathbf{z}_0 = \operatorname{argmin}_{\mathbf{z}} ||G_\phi(\mathbf{z}) - \mathbf{x}_0||$, and $\mathbf{x}_0$ is the original image.

2) Generative Networks: Each of these contrastive methods requires a generative model G_ϕ to reparameterize the input space. In this work, we test both a variational auto-encoder (VAE) and a generative adversarial network (GAN) as this generative network.

Each VAE used in this paper uses an encoder and decode of the following architectures. Each VAE encoder consists of 4 Conv2D layers with kernel size 3 and stride 1. The Conv layers output tensors with 16, 32, 64, and 128 channels. The last layer in the encoder is a linear layer that compresses the data into a 400-dimensional latent representation. The decoder of each VAE consists of a linear layer followed by 4 Conv2D layers. The linear layer goes from 400 to $28 \times 28 \times 128$, which is then unflattened to a tensor of width and height of 28 with 128 channels. The following encoders each have a kernel size of 3 with a stride of 1 and output 64, 32, 16, and 1 channels, respectively. Each layer uses 'same' padding to maintain the width and height. VAEs are trained using the Adam optimizer with weight decay 1e-4 and early stopping.

The generative adversarial networks used in this work are Wasserstein GANs with Gradient Penalty [28]. The WGAN generator has a latent dimension of 128 and consists of an initial ReLU-activated linear layer that is unflattened to a shape with a height and width of 4 and 256 channels. The rest of the network consists of 3 ConvTranspose2D that halve the number of channels in each layer while increasing the spatial dimension to 7×7, 14×14, and finally 28×28 with the last layer also reducing the number of channels to 1. Each of these layers has a kernel size of 4. The first two are ReLU activated, while the last is sigmoid activated. The discriminator consists of 3 sets of Conv2D with kernel size 5, output channels $64, 128, 256$, strides $2, 2, 1$, and padding 1 activated with LeakyReLU with slope 0.2. Each detector is trained using separate Adam Optimizers with a learning rate of 1e-4, $\beta_1 = 0$, and $\beta_2 = 0.9$. Training takes 400 epochs.

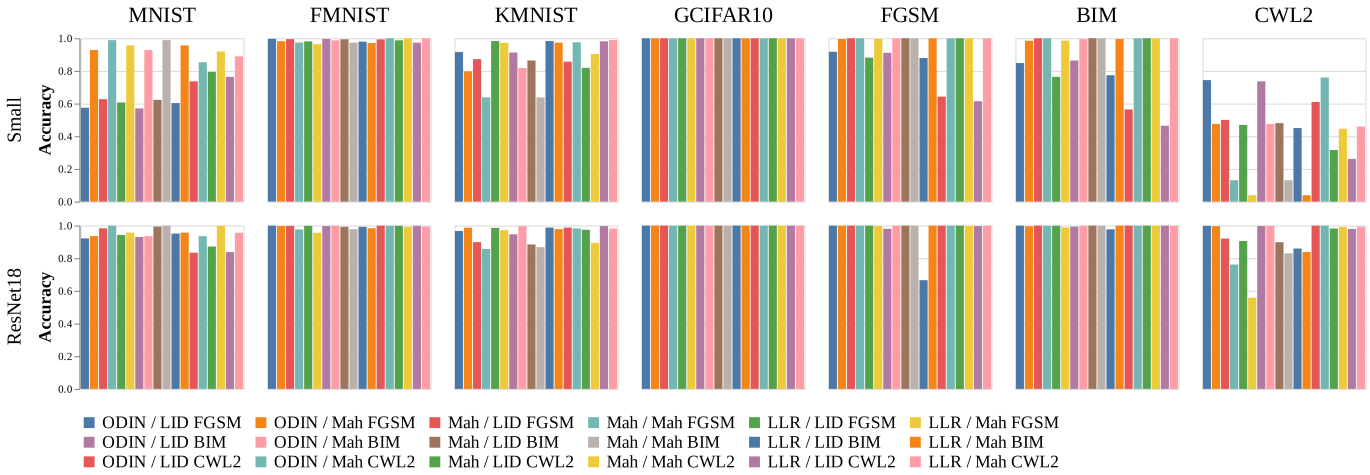

Fig. 2. Accuracy of each detection pairing on the in-distribution dataset (MNIST), OOD datasets (FMNIST, KMNIST, and Gray-CIFAR10), and adversarial datasets (FGSM, BIM, and CWL2). Each bar represents a pair of OOD and adversarial detectors, labeled as *OOD Detector/Adversarial detector Adversarial training set*. The top row provides the accuracy for each detector on the Small classifier while the bottom row provides the accuracy on the ResNet18 classifer.

3) Datasets: We conduct our experiments using MNIST, a dataset of handwritten digits, and two classification architectures from the literature. We closely follow Lee *et al.* [25] to create our training, validation, and test datasets. The training set of MNIST is used to train classifiers. We split the test set of MNIST into three partitions, with the first two partitions each containing 10% of the data and the last containing the other 80%. The first and second partitions are used for training (including validation) of the adversarial and OOD detectors, respectively. For the adversarial training set, we generate adversarial examples using each of the FGSM, BIM, and CWL2 methods. We additionally construct adversarial examples for the third partition to be used in testing the detectors. Only use data corresponding to successful adversarial attacks in both training and testing. For the OOD training set, we designate 10% of the partition to be used for training the detectors and 90% to be used for validation of the testers. Finally, we generate contrastive examples using the third partition.

4) Classifiers: The first architecture we use is similar to that used in LeCun *et al.* [29], consisting of two sets of Conv/MaxPool/ReLU, using a kernel size of 5×5 in the convolution, followed by two dense layers with the input resized to 64×64. We term this the *Small* architecture, as it is a basic architecture used to show how our method works. The second is a ResNet18 architecture [30] and is used to show how our method performs when the architecture scales.

We note that our experiments do not rely on any specific architecture, and others may be used as long as they function with contrastive generation and out-of-distribution and adversarial detection methods.

B. Joint Detection of OOD and Adversarial Examples

Well-functioning OOD and adversarial detectors are required to meaningfully approximate the DV of contrastive examples. As shown in Equation 7, it is possible for false negatives detected by the OOD or adversarial detector to be corrected by the other, but a false positive cannot be corrected. Due to this interaction in the performance of the overall detector, this first experiment evaluates the ability of a joint OOD and adversarial detection method to differentiate in-distribution data from that which is OOD or adversarial.

For this experiment, a classifier of each aforementioned architecture is trained on the MNIST dataset. Each of the ODIN [26], Mahalanobis [25], and Likelihood Ratio (LLR) [27] OOD detection models are trained for each classifier using the Fashion-MNIST (FMNIST) dataset as the OOD dataset. Local Intrinsic Dimensionality (LID) [24] and Mahalanobis adversarial detectors are trained on adversarial datasets constructed from FGSM, BIM, and CWL2 attacks applied to each classifier for a subset of the MNIST test set. We note that using these specific detection methods for both OOD and adversarial examples is not necessary, but they are instead selected to show the effects of various state-of-the-art techniques in our framework.

Both the adversarial and OOD Mahalanobis detectors were trained using noise magnitudes of 0, 0.0005, 0.001, 0.0014, 0.002, 0.0028, 0.005, and 0.01, using the best performance on the validation set. After the initial evaluation, we found that a temperature of 1000 was useful for the ODIN method and selected the best detector trained from noise magnitudes of 0, 0.0005, 0.001, 0.0014, 0.002, 0.0024, 0.005, 0.01, 0.02, and 0.05, using the best performing on the validation set. The LLR detector uses two VAE classifiers as described above. We have tested corruption magnitudes of 0.1, 0.2, 0.3, 0.4, and 0.5, finding that the most accurate detector came from using 0.3. We have trained LID detectors with k equal to 10, 20, 30, . . ., 90 and used the best performing one on the validation set.

Each joint detection method is evaluated on a separate test set of the training dataset (MNIST), the trained OOD dataset (FMNIST), and additional OOD datasets Kuzushiji-MNIST (KMNIST) and Gray-CIFAR10 that are used to test generality. Each joint detector is further evaluated on the test set of the

TABLE I

ACCURACY OF OUT-OF-DISTRIBUTION DETECTION METHODS (TOP-HALF) AND ADVERSARIAL DETECTION METHODS (BOTTOM-HALF) ON THE MNIST TEST SET, OUT-OF-DISTRIBUTION TEST SETS, AND ADVERSARIAL TEST SETS. EACH OOD METHOD WAS TRAINED USING MNIST AS THE IN-DISTRIBUTION DATASET AND FMNIST AS THE OUT-OF-DISTRIBUTION DATASET. EACH ADVERSARIAL DETECTION METHOD WAS TRAINED USING THE ADVERSARIAL DATASET INDICATED IN THE *Adv Method* COLUMN.

Classifier	Detector	Adv Method	MNIST	FMNIST	G-CIFAR10	FGSM	BIM	CWL2
Small	ODIN	-	0.9597	0.9660	0.9999	0.2013	0.1624	**0.3958**
Small	Mahalanobis	-	**0.9830**	**0.9754**	**1.0000**	**0.9972**	**0.8720**	0.1451
Small	LLR	-	0.9566	0.9394	0.9988	0.1996	0.0940	0.0382
ResNet18	ODIN	-	0.9353	**0.9934**	0.9999	0.9504	0.6127	**0.9699**
ResNet18	Mahalanobis	-	**1.0000**	0.9760	**1.0000**	**1.0000**	**0.9991**	0.7609
ResNet18	LLR	-	0.9566	0.9394	0.9988	0.2811	0.0600	0.0413
Small	LID	FGSM	0.6296	0.8591	0.9964	0.8513	0.7441	**0.4503**
Small	Mahalanobis	FGSM	**0.9998**	0.6711	0.9677	0.9947	0.9841	0.0009
Small	LID	BIM	0.6265	0.8484	0.9966	0.8519	0.7605	0.4295
Small	Mahalanobis	BIM	**0.9998**	0.8236	0.9969	0.9988	0.9939	0.0009
Small	LID	CWL2	0.7981	0.8176	0.9895	0.5089	0.4114	0.2316
Small	Mahalanobis	CWL2	0.9201	**1.0000**	**0.9999**	**1.0000**	**1.0000**	0.4440
ResNet18	LID	FGSM	0.9832	0.9924	**1.0000**	**1.0000**	0.9991	0.8981
ResNet18	Mahalanobis	FGSM	**0.9999**	0.8091	**1.0000**	0.9974	0.9872	0.5373
ResNet18	LID	BIM	0.9932	0.9052	0.9644	0.4940	0.9741	0.8475
ResNet18	Mahalanobis	BIM	**0.9999**	0.9761	**1.0000**	**1.0000**	0.9997	0.8294
ResNet18	LID	CWL2	0.8714	**0.9970**	**1.0000**	0.9947	0.9997	0.9774
ResNet18	Mahalanobis	CWL2	0.9989	0.9768	**1.0000**	**1.0000**	**1.0000**	**0.9907**

adversarial samples they were trained on as well as the test sets of the other two unseen attacks.

Figure 2 shows the results. For the Small classifier, the ODIN OOD detector and the Mahalanobis adversarial detector trained on the CWL2 attack, which we will refer to as the ODIN/Mahalanobis CWL2 detector, perform the best across the test sets. The detector classifies CWL2 attacks the best (76.0%) while maintaining high accuracy on the other test sets due to the ODIN OOD detector making up for shortcomings in the Mahalanobis detector trained on this classifier. Of the results that do not use the ODIN OOD detector, the Mahalanobis/Mahalanobis CWL2 and LLR/Mahalanobis CWL2 methods produce the best results, detecting 44.6% and 45.9% of CWL2 examples while achieving high accuracy on the in distribution and OOD test sets. It should also be noted that methods using LID adversarial detection trained on FGSM and BIM perform poorly at labeling MNIST data as in-distribution, leading to degraded performance.

For the ResNet18 classifier, many methods perform better at detecting in-distribution data and adversarial examples than for the Small model. In particular, the Mahalanobis/Mahalanobis CWL2 method performs best with high 90%-results on both the in-distribution and CWL2 test sets.

C. Out-of-Distribution and Adversarial Detector Individual Evaluation

In this experiment, we evaluate the ability of adversarial and out-of-distribution detectors to detect out-of-distribution and adversarial data, respectively, i.e., do the task the other type of detector was built to do. We train classifiers and detection methods as in Section IV-B. We evaluate each individual detection method on the in-distribution dataset MNIST, OOD datasets FMNIST and Gray CIFAR10, and adversarial datasets built with FGSM, BIM, and CWL2.

We present the accuracy of each detector in Table I. For the Small classifier, the ODIN OOD method is able to detect CWL2 attacks with accuracy on par of the other adversarial techniques. However, it is unable to distinguish the FGSM and BIM attacks well. On the other hand, the Mahalanobis OOD detection method is unable to distinguish CWL2 attacks well, but is able to pick up on FGSM and BIM attacks. We see this trend continue in the case of the ResNet18 classifier; however, the ODIN method is better able to detect FGSM attacks. The Likelihood Ratio method is unable to detect adversarial attacks well for either classifier, most likely due to having no dependence on the classifier in question. In most cases, the adversarial detectors were able to detect OOD examples much better than the OOD detectors were able to detect adversarial examples. In particular, the methods trained on CWL2 attacks had the best generalization.

D. Evaluating the Validity of Contrastive Example Generators

Using the best-performing detection methods found above, we now evaluate how well contrastive methods are able to generate DV examples. We compare the activation maximization (AM) method on the latent space, given by Equation 9, with xGEMs and CDeepEx, using VAE and GAN generators for the input reparameterization in each method. For each of these methods, we generate a contrastive dataset composed of inputs generated by the contrastive method for each label not equal to the label of the original image.

For each contrastive dataset, we test the target-class validity (TCV), in-distribution (ID) proportion, non-adversarial (NADV) proportion, and our devised DV proportion. We present the results with respect to the LLR/Mahalanobis CWL2 and ODIN/Mahalanobis CWL2 methods, due to their results in the previous section, in Table II.x

As shown in Table II, with respect to the LLR/Mahalanobis CWL2 detector and the Small network, the CDeepEx method

TABLE II

PERFORMANCE OF EACH CONTRASTIVE GENERATION METHOD WITH RESPECT TO THE LLR/MAHALANOBIS CWL2 AND ODIN/MAHALANOBIS CWL2 DETECTORS. THE METRICS DISPLAYED ARE: TARGET CLASS VALIDITY (TCV), IN-DISTRIBUTION (ID) PROPORTION, NON-ADVERSARIAL (NADV) PROPORTION, AND DISTRIBUTIONALLY VALID (DV) PROPORTION.

			LLR/Mah CWL2			ODIN/Mah CWL2		
	Method	TCV	ID	NADV	DV	ID	NADV	DV
Small	AM VAE	**1.0000**	0.1456	0.0344	0.0233	**1.0000**	0.0344	0.0344
	AM GAN	0.9700	0.6944	0.0167	0.0067	0.9989	0.0167	0.0067
	xGEMs VAE	**1.0000**	0.9667	0.2022	0.1978	0.1056	0.2022	0.0278
	xGEMs GAN	**1.0000**	0.9044	0.7311	0.6700	0.1900	0.7311	**0.1567**
	CDeepEx VAE	0.9956	**0.9722**	0.2522	0.2500	0.0622	0.2522	0.0267
	CDeepEx GAN	0.9933	0.9011	**0.7511**	**0.6889**	0.0878	**0.7511**	0.0656
ResNet18	AM VAE	0.9067	0.4189	0.1156	0.0678	**0.9911**	0.1156	0.0722
	AM GAN	0.7533	0.7400	0.2967	0.1567	0.9633	0.2967	**0.1633**
	xGEMs VAE	**0.9600**	0.9622	0.7544	0.7122	0.0244	0.7544	0.0167
	xGEMs GAN	0.9511	0.8833	0.9256	**0.7822**	0.0311	0.9256	0.0300
	CDeepEx VAE	0.5533	**0.9367**	0.7989	0.4222	0.2444	0.7989	0.1522
	CDeepEx GAN	0.3811	0.9067	**0.9367**	0.3200	0.0567	**0.9367**	0.0278

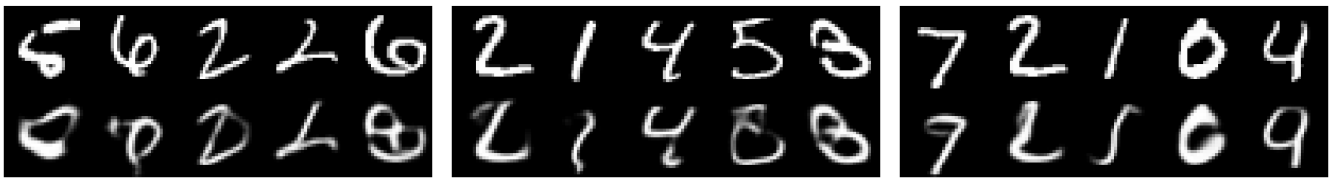

Fig. 3. Samples judged to be either OOD (left), adversarial (middle), or distributionally valid (right) by the LLR/Mahalanobis CWL2 detection method. Contrastive examples were generated using the xGEMs method with a VAE generator for the ResNet18 classifier. Each panel contains a set of the original images in the top half of the panel and a set of the corresponding contrastive examples in the bottom half. The contrastive labels assigned by the classifier in the adversarial (middle) panel from left to right are 6, 8, 0, 0, 6. The contrastive labels assigned by the classifier for the DV (right) samples from left to right are 9, 1, 5, 6, 9.

using a GAN generative network gave the best Valid and Non-Adversarial proportions, while for the ResNet18 network, xGEMs using a GAN generative network gave the best DV score. CDeepEx has a significantly lower DV score for the ResNet18 model due to low target-class validity. This comes from the optimization objective of CDeepEx, which tries to find the classifier's decision boundary between the original and target classes. Additionally, for both classifiers, both generative networks, and detection methods, the AM method performed worse than both xGEMs and CDeepEx. We speculate that this comes from optimizing the $\log p(w_{y'}|G_\phi(\mathbf{z}))$ term of Equation 9 without regard for the other classifier logits. This results in OOD and adversarial examples that are additionally stable, but not necessarily minimal.

The results using ODIN as the OOD detector do not tell the same story as that described above. ODIN uses a preprocessing step that pushes the data along the error gradient given the class predicted by the classifier, which can help distinguish OOD data from in-distribution data. However, this preprocessing step assumes that data is locally stable, i.e., data in close proximity produce similar classifications, which is not the case for contrastive examples that lie near the decision boundary of a classifier. Therefore, the ODIN/Mahalanobis CWL2 method artificially boosts the reported proportion of in-distribution contrastive examples output by the AM method, because the AM method produces very stable, if incorrect, classifications by optimizing specific logits output by the classifier. It also artificially lowers the reported proportion of in-distribution contrastive examples output by the xGEMs and CDeepEx methods, as these methods provide contrastive examples that are closer to the decision boundary of the classifier.

We can also observe slight fluctuations in TCV values across the settings of the Small network, which occurs despite the non-negligible variations of the OOD and adversaial examples. Similarly, the TCV scores related to ResNet18 do not entirely align with the distribution of OOD and adversaial examples produced by these contrastive example generation methods.

In addition to the above quantitative results, we provide sample contrastive examples generated by xGEMs with a VAE applied to the ResNet18 classifier that were labeled as OOD, adversarial, or DV in Figure 3. These images show that, as predicted, the adversarial detector makes up for error in the OOD detector, removing some OOD data at this step.

V. CONCLUSION

In this paper, we propose a refinement to the validity property of contrastive examples based on classifier agreement with the data distribution, namely distributional validity, which is important to accurately provide recourse to those affected by automated decision systems. We evaluate contrastive generation methods using this property and find that the best generation method can vary on the architecture of the employed classifier. This work introduces theoretical results in image space, a highly structured and correlated data space, but could be applied to other domains, including tabular datasets, in order to better provide recourse automated

system users. Finally, in this study we conduct our experiments using automated methods. Follow up experiments using user studies would allow us to understand how other interpretability metrics are correlated with distributional validity.

ACKNOWLEDGMENT

This research has been sponsored by the National Science Foundation through grant IIS-1652846. This work was completed utilizing the Holland Computing Center of the University of Nebraska, which receives support from the UNL Office of Research and Economic Development and the Nebraska Research Initiative.

REFERENCES

[1] M. Bakator and D. Radosav, "Deep learning and medical diagnosis: A review of literature," *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 47, 2018.

[2] K. Freeman, "Algorithmic injustice: How the wisconsin supreme court failed to protect due process rights in state v. loomis," *North Carolina Journal of Law & Technology*, vol. 18, no. 5, p. 75, 2016.

[3] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)," *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.

[4] A.-H. Karimi, G. Barthe, B. Schölkopf, and I. Valera, "A survey of algorithmic recourse: contrastive explanations and consequential recommendations," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–29, 2022.

[5] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, and C. Shah, "Counterfactual explanations and algorithmic recourses for machine learning: A review," *arXiv:2010.10596*, 2022.

[6] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach, "FACE: Feasible and actionable counterfactual explanations," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 344–350.

[7] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[8] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 607–617.

[9] S. Joshi, O. Koyejo, B. Kim, and J. Ghosh, "xGEMs: Generating examplars to explain black-box models," *arXiv:1806.08867*, 2018.

[10] A. Feghahati, C. R. Shelton, M. J. Pazzani, and K. Tang, "CDeepEx: Contrastive Deep Explanations," in *ECAI 2020*, 2020, pp. 1143–1151.

[11] A. Van Looveren and J. Klaise, "Interpretable Counterfactual Explanations Guided by Prototypes," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, vol. 12976 LNAI, 2021, pp. 650–665.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.

[13] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[14] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[15] A. Joshi, A. Mukherjee, S. Sarkar, and C. Hegde, "Semantic Adversarial Attacks: Parametric Transformations That Fool Deep Classifiers," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 4772–4782, apr 2019.

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

[17] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[18] T. Laugel, M.-J. Lesot, C. Marsala, and M. Detyniecki, "Issues with post-hoc counterfactual explanations: a discussion," in *ICML Workshop on Human in the Loop Learning (HILL 2019)*, 2019.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[20] Z. Zhang, K. Qiao, L. Jiang, L. Wang, J. Chen, and B. Yan, "AdvJND: Generating Adversarial Examples with Just Noticeable Difference," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12487 LNCS, pp. 463–478, feb 2020.

[21] A. Kurakin, I. Goodfellow, S. Bengio *et al.*, "Adversarial examples in the physical world," 2016.

[22] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[23] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017.

[24] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[25] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[26] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[27] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[28] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.