CDeepEx: Contrastive Deep Explanations

Amir Feghahati and **Christian R. Shelton** and **Michael J. Pazzani** and **Kevin Tang** [1]

Abstract. We propose a method which can *visually* explain the classification decision of deep neural networks (DNNs). Many methods have been proposed in machine learning and computer vision seeking to clarify the decision of machine learning black boxes, specifically DNNs. All of these methods try to gain insight into why the network "chose class A" as an answer. Humans search for explanations by asking two types of questions. The first question is, "Why did you choose this answer?" The second question asks, "Why did you *not* choose answer B over A?" The previously proposed methods are not able to provide the latter directly or efficiently.

We introduce a method capable of answering the second question both directly and efficiently. In this work, we limit the inputs to be images. In general, the proposed method generates explanations in the input space of *any* model capable of efficient evaluation and gradient evaluation. It does not require any knowledge of the underlying classifier nor use heuristics in its explanation generation, and it is computationally fast to evaluate. We provide extensive experimental results on three different datasets, showing the robustness of our approach, and its superiority for gaining insight into the inner representations of machine learning models. As an example, we demonstrate our method can detect and explain how a network trained to recognize hair color actually detects eye color, whereas other methods cannot find this bias in the trained classifier.

1 Introduction

Deep neural networks (DNN) have shown extraordinary performance on computer vision tasks such as image classification [34, 32, 33, 10], image segmentation [5], and image denoising [38]. The first example of such a performance was on image classification, where it outperformed other computer vision methods which were carefully handcrafted for image classification [18]. Following this success, DNNs continued to grow in popularity. Even with DNNs achieving testing accuracy close to human expertise [26] and in some cases surpassing them [33], there is hesitation to use them when interpretability of the results is important. Accuracy is a well-defined criterion but does not provide useful understandings of the concept captured by the network. If the deployment of a network may result in inputs whose distribution differs from that of the training or testing data, interpretability or explanations of the network's decisions can be important for securing human trust in the network.

Explanations are important in settings such as medical treatments, system verification, and human training and teaching. Naturally, one way of getting an explanation is asking the *direct* question, "Why did the DNN choose this answer?" Yet, humans often also seek *contrasting* explanations. For instance, they may be more familiar with the contrasting answer, or they want to find the subtle differences in input which change the given answer to the contrasting one. This way of questioning can be phrased as, "Why did the DNN *not* choose B (over A)?" In this work, we present a framework to answer this type of question.

To present the explanation in the space of *natural images*, with which humans are accustomed, we assume a generative model on the input space exists. This model, which is trained on the input space and not necessarily on the input dataset of the DNN, is able to provide synthetic samples similar to the input. Then, we ask how we can alter this *synthetic* input to change the classification outcome. Our proposed framework is not based on heuristics, does not need to change the given network, is applicable as long as the given model can handle backpropagation (no further requirements for layers), and can run much faster than methods with input perturbation. The only overhead of this method is the assumed availability of a latent model over the input. If this latent model is not available, we can learn such a model using a generative adversarial network (GAN) or variational auto encoder (VAE). Learning this latent space needs to be done only a single time and is independent of the learned classifier to be explained.

2 Related Work

Some past work has mapped queries back to the training set of the examined network [17, 19]. This can be effective in providing guidance on dataset changes. However, training methods or network architectures may also be responsible for the network's effects. Further, the resulting network is its own object and its decisions can be explain irrespective of how it was trained to produce them. To that end, our work and the work we describe below treat the network as a given and seek explanations of the function given, not of the learning method that produced the function.

Our problem is distinct from work on adversarial examples in that we are not interested in how to break the classifier, but rather what concept the classifier has learned. For instance, an adversarial example which adds small amounts of speckled noise is not helpful for understanding the concept *in the space of natural image variation*. This drives our use of latent-space representation of the input (through GANs or VAEs). There are uses of interpretations to secure classifiers against adversarial examples. However, our focus is on non-adversarial explanations.

There are different ways to categorize interpretability methods [30]. Here we categorize the existing approaches into three overlapping categories, focused on methods for deep neural networks.

2.1 Network Visualizers

The first group of methods try to understand units of the network [7, 36, 2]. These methods test each individual network unit or a set of units to gain insight into what network has learned.

[1] University of California, Riverside, USA, {sfegh001,cshelton,pazzani,ktang012}@ucr.edu

The disadvantage of these methods is that they need to check all the units to see which one (or combination of units) is responsible for a concept. The method proposed in [22] showed it is unlikely that only a single unit learns a concept. Rather, a set of units usually combine to represent a concept. This, in turn, makes these methods inapplicable in practice when the network contains thousands of units. These methods are example-based explanations. That is, they generate an explanation for a single input. By contrast, [8] proposed a method to determine whether the network learned a concept based on a set of probe images and pixel-level annotated ground truth which may not be readily available or easy to obtain for many tasks.

2.2 Input Space Visualizers

The second category corresponds to networks that try to explain the network's decision in the space of the input image. Authors of [27] proposed a method to find out which parts of the image have the largest contribution to the decision of the network by making changes to the image and forwarding the new image through network. The method by [40] proposed a similar approach with a more clever way of sampling the image parts. These methods need to consider changing each dimension of the image and find an explanation for each change in order that the aggregated results are visually coherent. In [6] a method proposed which also takes into account the middle layers of the network to produce the explanation. [39] proposed a method which forwards the image through the network, records the activations in the last pooling or convolution layer, and uses them as an explanation. [29] propose a similar method which uses the back-propagated signal for a more accurate explanation. There are two potential difficulties with these approaches. First, they assume that the explanation can be summarized in the last layer which has a broad receptive field which may not be appropriate for fine-grained datasets. Second, these methods are restricted to use in convolutional networks.

[31] used the gradient of the output with respect to the pixels of the input to generate a heat map. They showed that their proposed method is closely related to DeconvNets [37]. The difference is in the handling of backpropagation of ReLU units. The weakness of these methods is that the generated backpropagated signal in image space is not visually specific. They need to use heuristics in backpropagation to make the results more specific and useful to humans, such as changing the backpropagation rules for ReLU unit [33]. These methods have been shown to be unreliable with respect to shifts in the input [15]. Some of these methods need a reference input image [30] whose choice can greatly change the generated explanation [15]. In [13], authors discussed the issue of learning a surface statistics rather than a high level concept. In 4 Section, we examine that how much of this claim might be true. [23] propose a method to find a sparse combination of filter responses which are the most responsible for the network's classification. [4] explores how the network behavior changes if some part of the image changes based on a Bernoulli prior, but do not create a real counter-factual as we do.

The most similar work to ours is the unpublished xGEMs [14], a preprint available on arXiv. xGEMs is similar to our work in that it uses a GAN to regularize the explanation and also seeks to find a "why not" explanation. Their work is different in that it does not formulate a constrained optimization problem as we do (our results show the importance of our constraints), and they focus on producing a "morph" from one class to another, rather than highlighting the differences directly in the input image (as we do).

Input space visualizers (including our own) produce one example of how the input might be changed to change the classification or output of the network being analyzed. If the demonstrated change does not match the human's understanding of the task, this example explains a problem with the classifier. However, there may be other examples of input changes which similarly modify the classifier's output. Therefore, if the demonstrated change matches the human's understanding of the task, this does not guarantee that there are not other undesirable properties of the analyzed network.

2.3 Justification Explanations

Finally, there are methods that learn to justify the classification of a network by producing textual or visual justifications from training explanations. [11, 24]. Although related to image descriptions or definitions, they differ in that their goal is to explain *why* in a general sense. Because the justification network is trained to match human explanations, they are not direct explanations of how the learned classification network makes its decisions, but rather what humans would like as the explanation. If the explanation is "correct," we cannot be certain this is because the network learned the concept correctly, or because the explanation network learned to mimic what we want to hear.

2.4 Summary

Existing methods have one or more of these downsides:

I They are only applicable to specific architectures. [39, 29]
II They use heuristics during backpropagation. [33, 37]
III They need of a set of probe images or concepts. [30]
IV They need network alteration to record the activations. [37, 2, 30]
V They need considerable computational time. [40, 27]
VI They learn an explanation, and are trained to produce the explanations we desire, rather than faithful ones. [24]

We have replaced these downsides (assumptions, heuristics, probe images, or other black-boxes) with a generative latent-space model (built with a GAN or VAE). We submit that this latent-space model imposes less bias on the explanations, while still providing a human-understandable explanation. The unbiased, true explanation for any classification is the (long) sequence of calculations performed. For human understanding, these must be filtered through some type of lens.

We believe the latent input space is a natural bridge between the network's understanding and the human's understanding, as it is common data (or language) to both, unlike raw pixels or activation values (which are natural for the network, but not for the human). Other bridges, like natural language or scene objects, require assumptions about how the network works (which call into question whether the explanation is valid) or training up other networks to make the bridge. While we also require another network, there is less bias imposed by modeling the space of natural images than by modeling the mapping from neural network concepts to natural language or other higher-level concepts.

Our lens, of the latent-space parameterization of the input domain, can be trained on completely different data than that used for the classifier to be analyzed (Section 4.6), can be shared across multiple classifiers with the same input domain (thus providing a standard testing tool), and can be improved and tested externally and independently from the classifier-to-be-analyzed. We further show experiments demonstrating that the generative model does not project its own training set (Section 4.2) onto the explanations for a network trained on different data.

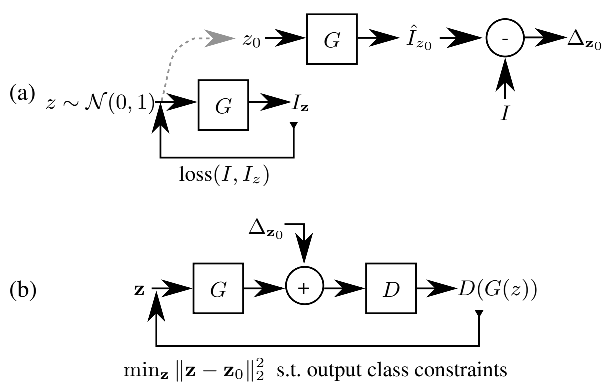


Figure 1: The schematics of the proposed approach. (a) First, in the case of using a GAN, find $\mathbf{z}_0$ which gives $G(\mathbf{z}_0) \approx \mathbf{I}$. (If using a VAE, the $\mathbf{z}_0$ that best approximates $\mathbf{I}$ can be generated directly.) Then, let $\Delta_{\mathbf{z}_0}$ to be the difference between the input image $\mathbf{I}$ and reconstructed one, $G(\mathbf{z}_0)$. (b) Last, generate the final explanation by optimization over z. $\mathbf{z}_0$ and $\Delta_{\mathbf{z}_0}$ are fixed.

3 Proposed Method

First, we introduce the notation used in this work. Then, we describe how to learn a latent space capable of generating natural looking images similar to the input space. Last, we describe our method on how to generate explanations from the latent representation of input space. The overall framework is summarized in Algorithm 1 and Figure 1.

3.1 Terminology

$D : R^n \to R^c$ is the given and fixed discriminator network, for which we want to generate explanations. $G : R^k \to R^n$ generates natural looking images in the input domain. It should be able to generate images similar to those being queried, but need not be directly related to D. $\mathbf{I} \in R^n$ is the input image; $z \in R^k$ is a latent variable; and $\mathbf{I}_{\mathbf{z}}$ is the output of G, i.e. $\mathbf{I}_{\mathbf{z}} = G(z)$. We define y_{true} as the label produced by D for image I, and y_{probe} as the class label for the class of interest. Note that y_{true} may not be the true label from an accuracy point-of-view, but it is the true label produced by the discriminator network D (compared with y_{probe} which is a counter-factual label not actually produced by D).

Thus, the question we would like to answer is "Why did D produce label y_{true} and not label y_{probe} for input $\mathbf{I}$?"

Algorithm 1 Generating the explanation on given D and $\mathbf{I}$

```
1: Learn a function $G : \mathbb{R}^k \to \mathbb{R}^n$          ▷ if necessary
2: Find a representation for input $\mathbf{I}$ using Algorithm 2
3: Find $\mathbf{z}_e$ from Equation 2
4: Return the explanation $G(\mathbf{z}_0) - G(\mathbf{z}_e)$
```

Algorithm 2 Generate the latent representation on the input $\mathbf{I}$

```
1: procedure LEARN $\mathbf{z}_0$ $(G, \mathbf{I}, \eta, \mathrm{loss}(.))$
2:     $\mathbf{z}_0 \sim \mathcal{N}(0,1)$
3:     while $G(\mathbf{z}_0) \not\approx \mathbf{I}$ do
4:         $\mathbf{z}_0 \leftarrow \mathbf{z}_0 - \eta \nabla_{\mathbf{z}} \mathrm{loss}(\mathbf{I}, G(\mathbf{z}))$
5:     $\Delta_{\mathbf{z}_0} = G(\mathbf{z}_0) - \mathbf{I}$
6:     return $\mathbf{z}_0, \Delta_{\mathbf{z}_0}$
```

3.2 Learning the Input Distribution

The question "why not class y_{probe}?" implies a query image about which the question is being asked. We need to capture the manifold of natural looking images similar to this input to be able to answer this question in a meaningful way for a human. Learning a compact manifold enables us to move along this manifold instead of directly optimizing in the input space of raw pixels.

There are different ways to find this mapping including variational auto encoders (VAEs) [16] and generative adversarial networks (GANs) [9]. In this work, we use both GANs and VAEs to map latent space into input space. We used the method proposed by [1] to train the GAN. The structure of the networks is similar to that proposed by [25]. It worth noting that unlike [19], the generated explanation must not include in the training set. We want to create explanation about the black-box model itself and not how it was created.

3.3 Generating Explanations

First, we need to find an initial point in the latent space which represents the input image, i.e., $G(\mathbf{z}_0) \approx \mathbf{I}$. If G was generated by a VAE, this can be done by feeding $\mathbf{I}$ into the encoder half of the VAE. If G was generated by a GAN, we find this initial point by solving $\mathbf{z}_0$ as

$$\mathbf{z}_0 = \arg\min_{z} \ \mathrm{loss}(G(z), \mathbf{I}) \tag{1}$$

in which $\mathrm{loss}(.)$ is a suitable loss function, e.g. $||.||_2$ distance for images.

Since the generated image $G(z)$ may be classified into a different class by the discriminator, we add a bit of the misclassification cost for $G(z)$ to our loss function to insure that the resulting image not only looks similar, but is also classified the same way as $\mathbf{I}$ by D. We only add this extra cost for the GAN generators. As the final fit will not be exact, we define $\Delta_{\mathbf{z}_0}$ to be the residual error: $\Delta_{\mathbf{z}_0} = G(\mathbf{z}_0) - \mathbf{I}$. This residual makes the latent space generated image exactly equal to the input image. See Algorithm 2 for more details.

Next, we find a change in latent space for which the change in the input space explains why, for this particular image, class y_{probe} is not the answer. In particular, we seek the closest point (in the latent space of $\mathbf{z}$) for which y_{probe} would be equally likely as y_{true}, and all other classes are less probable. This is a point which is most like the input, yet would be class y_{probe}. Thus, the answer to the question is, "It is *not* like *this*."

To do so, we solve a constrained optimization problem:

$$\begin{aligned} \mathbf{z}_e = \quad & \arg\min_{\mathbf{z}} ||\mathbf{z} - \mathbf{z}_0||_2^2 \\ \text{s.t.} \quad & \mathrm{llh}(D(\mathbf{I}_{\mathbf{z},\mathbf{z}_0}), y_{\text{true}}) - \mathrm{llh}(D(\mathbf{I}_{\mathbf{z},\mathbf{z}_0}), y_{\text{probe}}) = 0 \\ & \mathrm{llh}(D(\mathbf{I}_{\mathbf{z},\mathbf{z}_0}), y_{\text{true}}) - \mathrm{llh}\left(D(\mathbf{I}_{\mathbf{z},\mathbf{z}_0}), y'\right) \geq \epsilon \\ & \mathrm{llh}(D(\mathbf{I}_{\mathbf{z},\mathbf{z}_0}), y_{\text{probe}}) - \mathrm{llh}\left(D(\mathbf{I}_{\mathbf{z},\mathbf{z}_0}), y'\right) \geq \epsilon \\ & \quad \forall y' \neq y_{\text{true}}, y' \neq y_{\text{probe}} \end{aligned} \tag{2}$$

in which $\mathbf{I}_{\mathbf{z},\mathbf{z}_0} = G(\mathbf{z}) + \Delta_{\mathbf{z}_0}$ and $\mathrm{llh}(f, y)$ is *log-likelihood* of class y for classifier output f. Our visual explanation is the difference between $\mathbf{I}$ and $\mathbf{I}_{\mathbf{z}_e,\mathbf{z}_0}$.

The first constraint forces the explanation to be on the boundary of the two classes. However, because D is complex, this can lead to solutions in which the likelihood of y_{probe} and y_{true} are equal, but another class has an even higher likelihood. To overcome this, we impose the last two constraints which enforce that the class produced by D and the probe class remain the most likely. We further illustrate the necessity of these constraints in Section 4.

3.3.1 *Optimization Method*

Using the method of augmented Lagrangian multiplier, we can convert the constrained optimization problem into a series of unconstrained ones. Complete details of the optimization procedure are discussed by [3]. In summary, we convert Equation 2 into the augmented Lagrangian

$$
\begin{aligned}
L_c(\mathbf{z}, \lambda, \mu) = {} & \|\mathbf{z} - \mathbf{z}_0\|_2^2 \\
& + \lambda^T h(\mathbf{z}, y_{\text{true}}, y_{\text{probe}}) + \frac{c}{2}\|h(\mathbf{z}, y_{\text{true}}, y_{\text{probe}})\|_2^2 \\
& + \frac{1}{2c}\sum_{\substack{y' \neq y_{\text{true}} \\ y' \neq y_{\text{probe}}}} \left\{ \left(\max\left\{0, \mu_{y'} + ch(\mathbf{z}, y_{\text{true}}, y')\right\}\right)^2 - \mu_{y'}^2 \right\} \\
& + \frac{1}{2c}\sum_{\substack{y' \neq y_{\text{true}} \\ y' \neq y_{\text{probe}}}} \left\{ \left(\max\left\{0, \tilde{\mu}_{y'} + ch(\mathbf{z}, y_{\text{probe}}, y')\right\}\right)^2 - \tilde{\mu}_{y'}^2 \right\}
\end{aligned} \tag{3}
$$

where

$$
h(\mathbf{z}, y_1, y_2) = \text{llh}\left(D(\mathbf{I}_{z,z_0}), y_1\right) - \text{llh}\left(D(\mathbf{I}_{z,z_0}), y_2\right) = 0
$$

and λ, $\mu_{y'}$, and $\tilde{\mu}_{y'}$ are Lagrange multipliers for the constraints of Equation 2. We solve the constrained optimization problem by solving a series of unconstrained optimizations of Equation 3 for a series of c values. Each individual optimization we solve with standard gradient descent. The update rule for c is

$$
c^{k+1} = c^k \beta^{\mathbb{1}\left(\|h(\mathbf{z}^k, y_{\text{true}}, y_{\text{probe}})\| > \gamma \|h(\mathbf{z}^{k-1}, y_{\text{true}}, y_{\text{probe}})\|\right)} \tag{4}
$$

in which $\mathbb{1}$ is the indicator function, γ is 0.24, β is 1.01 and initial value of c is 1.

4 Experimental Results

We compared our method, contrastive deep explanation (CDeepEx), with those of [27, 29, 40, 28]. Some of these methods try to answer the question "why class A?" instead of our contrastive question. However, it would be natural to try use the difference of the answers to "why class A?" and "why class B?" as a contrastive answer. Thus, we compare to them to demonstrate the need for a different method to answer these contrastive questions.

First, we tested these methods on two datasets: MNIST [20] and fashion-MNIST [35]. Although MNIST seems to be a very simple dataset, it has its own challenges when it comes to contrasting two outputs. This dataset has ambiguities that are hard even for humans to identify. We also expect the reader to has prior knowledge of the digits (compared to, perhaps, birds or skin cancers), thus making the results easier to interpret. We conducted experiments using two different generative models (GANs) and Variational Auto Encoders (VAEs). We show that our method works well regardless of the choice of the generator.

4.1 MNIST

In this section, we find explanations for contrasting categories using our method (CDeepEx), Lime [27], GradCam [29], PDA [40], LRP [28] and xGEMs [14]. The network architecture for D is similar to that used by the original MNIST papers, consisting of two sets of Conv/MaxPool/ReLU layers following by two fully connected layers. The input is resized to 64x64. The kernel size for each convolution layer is 5x5 and the first fully connected layer is 3380x50.

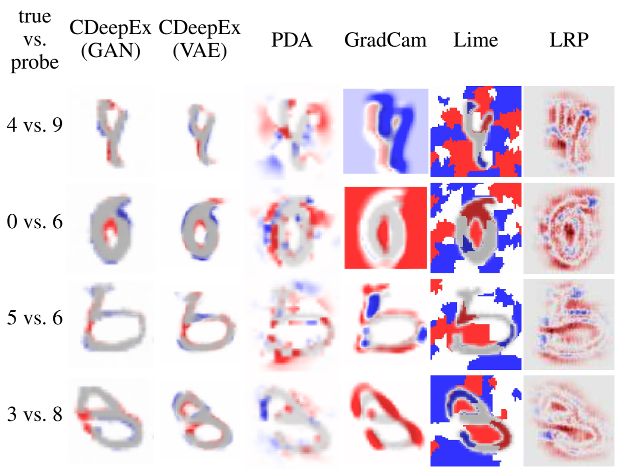


Figure 2: Generated explanations for MNIST for examples correctly classified by the network. The input image is in gray. Red indicates regions that should be added and blue regions that should be removed (to move from the true label to the probe label). Thus, the absence of the red regions and the presences of the blue regions is the explanation.

GradCam, Lime and LRP were not designed to answer this type of question directly. However, these methods could be shoe-horned into trying to answer the question of"why A and not B?" and so we figured we should demonstrate that they were not sufficient and that a new method (like ours) was necessary. Therefore, we generate the explanation by first extracting the explanations for the true and probe classes. Then we assign different weights to each of these explanations and subtract them from each other. If imposing this explanation on the input image decreases the network probability for the true class and increases it for the probe class, we keep this explanation. The weights for the probe explanation we used are from 0.005 to 2 with increments of 0.005. The final explanation is the average of all the maps that satisfies the mentioned condition. We tried multiple other methods to contrast the explanations of the true and probe classes and this method produced the best and most reliable results, although clearly GradCam and Lime were not designed for this scenario.

GAN structure follows the guidelines of [25]. It has 5 set of ConvTranspose2d/BatchNorm2d/ReLU operations. The size of $\mathbf{z}$ is 100. The VAE has four convolutional layers and two fully connected layers, one for μ and one for σ. The size of the latent code is 400 for each of the mean and standard deviation.

4.1.1 *General Comparisons*

Figure 2 shows explanations generated using CDeepEx, PDA, GradCam, Lime and LRP. We trained the discriminator on the unmodified MNIST dataset with a resulting success rate of 99% in testing phase. For testing examples, we asked why the output was not the second-highest likelihood class. (That is, the probe class is the second-most likely class, according to the classifier.)

The first example shows why D believes the class to be a 4 and not a 9. Our method shows that this is because the gap on the top of the digit is not closed. The second row is the explanation for why a 0 and not a 6. The only meaningful explanation is generated from CDeepEx: because the circle is thicker and the top extending line is not more pronounced. The third row is for why a 5 and not a 6. Although Lime shows that the opening gap would have to be closed, it produces many other changes that detract from the central reason.

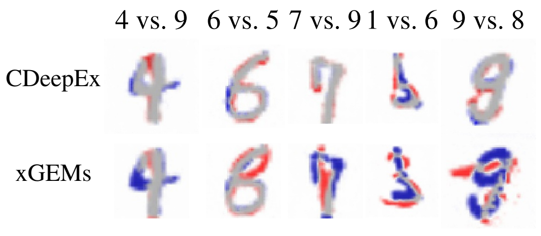


Figure 3: Comparison of CDeepEx to xGEMs. Colors are as in Figure 2. Please check Figure 5 for more qualitative results.

For the last row, the only correct explanation for why a 3 and not an 8 comes from our method. Note that GradCam and Lime were not designed for this type of explanation, but we tried hard to find the best way to make contrasting explanations from their single-class explanations (see above).

4.1.2 Comparisons to xGEMs and Optimization

In the next experiment, we show the importance of having the constraints in the optimization and how it affects the explanation compared to xGEMs. In Figure 3, we compare our results with our implementation of xGEMs. The primary difference between the methods is the last two sets of constraints in Equation 2 (which are present in our method, but lacking from theirs). This figure shows that without the constraints, the explanation are not correct or are of worse quality.

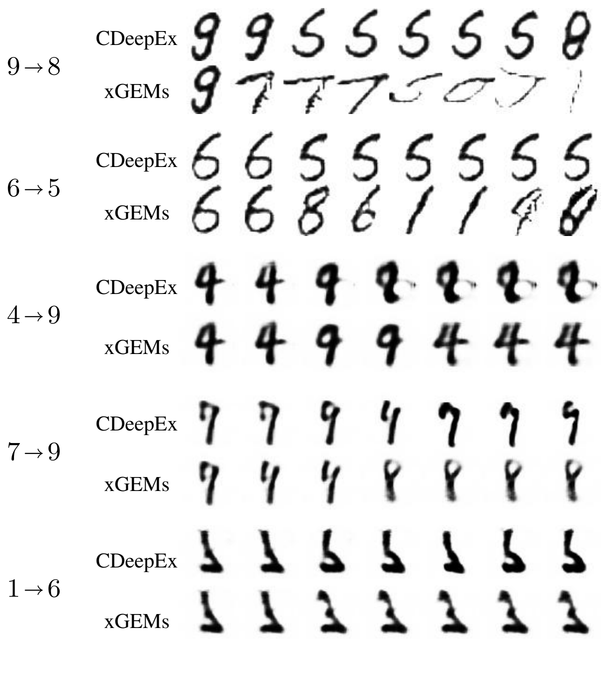
In Figure 4 we show the optimization path to find the explanation, with (CDeepEx) and without (xGEMs) constraints. The program has a non-linear objective with non-linear constraints, so the path is particularly important, as we will not (generally) find the global optimum. Without the constraints, the found $\mathbf{z}$ results in equal likelihood for y_{true} and y_{probe}, but a third class consistently has even higher likelihood, thus generating an explanation more suitable to this third class than the requested probe class. This is typical for xGEMs and explains the xGEMs row of Figure 3, where frequently a third random class is superimposed on the explanation.

4.1.3 Generator Model and Starting Point

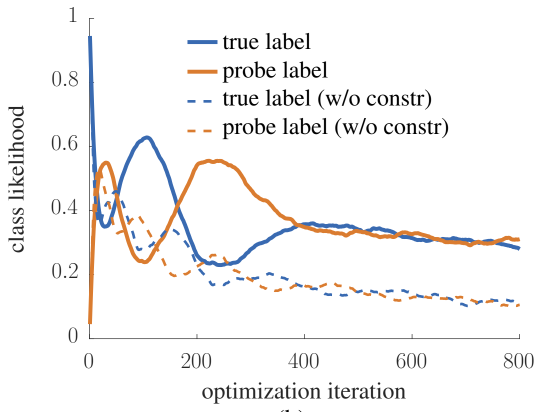
In this section, we experiment with different generator models and different starting points for GANs to see if the results change significantly or not. We used a GAN and a VAE as examples of these generator models. In Figure 5 we show that for three different $\mathbf{z}_0$ as starting points for GANs, our results are visually consistent for most of the inputs while there are inconsistencies for xGEMs method. We believe the source of the inconsistencies are: a) $\Delta\mathbf{z}_0$ can be visually noticeable for GANs, and, b) the nature of the GAN itself. It worth noting that $\Delta\mathbf{z}_0$s in our experiments are so minuscule that a human cannot distinguish between the original image and the synthetic image. As it is clear, we do all of our experiments in such a datasets that can be seen as fine-grained classification. For instance, changing a 0 to an 8, is just adding a horizontal line in the middle of the image. By contrast, we are not interested on producing explanations on dataset such as ImageNet since changing an instance of class chair to a dog consists of entirely changing the image.

4.2 Selection of the Generator Model

Although we showed the results on three different datasets using AE (see 4.4), VAE and GAN, one might argue that the explanation



(a)



(b)

Figure 4: Examples of the optimization path for Equation 2. CDeepEx is our method. xGEMs is similar, but without the last pair of constraints. The optimization goal is to find a $\mathbf{z}$ such that $G(\mathbf{z})$ is maximally confused between the two classes. (a) Examples from $G(\mathbf{z})$ along the optimization path of $\mathbf{z}$, using a GAN. (b) Average class likelihood from $D(G(\mathbf{z}))$ for all examples (probe is second-most-likely class). Both demonstrate that without the constraints (xGEMs), the optimization finds an example for which the true and probe likelihoods are equal, *but* another class has an even higher likelihood. Our method with the constraints keeps the explanation targeted to the true and probe classes.

solely comes from the generator network. To address this, we designed another experiment, to show the explanation comes from the discriminator network and not the generative model.

Using the MNIST dataset, first, we train the network without showing it any images from class "8." This drops the testing accuracy on that class to 0. Then, we show some samples of the class "8" to the network, increasing its accuracy on class "8" to 0.3. We repeat this procedure by showing more and more samples to the network, saving its parameters at 0.6 and 0.99 accuracy. Then, we run experiments, using the same generator network G, asking why D did not choose

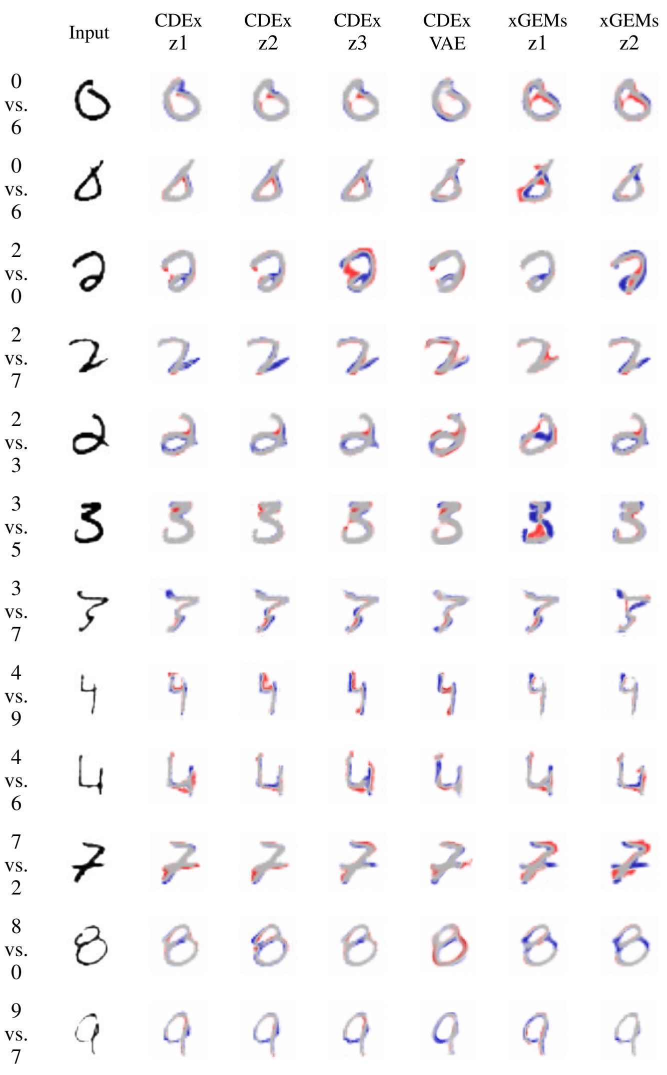


Figure 5: Additional experiments comparing our method using a GAN or a VAE with xGEMs. The multiple columns for the GAN methods are for different random starting points for $\mathbf{z}_0$.

the class "8." The results are shown in Figure 6. While the same latent-space representation is used, the explanations are particular to the network trained.

4.3 Biased MNIST

To test to see if our method could provide clear explanations of the bias of a classifier, we trained a network with the same structure as the previous experiments, but on modified data. In this experiment we added a 6x6 gray square to the top left of all the images for class "8." If tested on the true data (without the modifications), the network only recognizes 5% of the 8s correctly. Also, if we add the square to all the testing images, the network recognized 77% of other classes as "8."

We then compared our method to others in explaining the predictions of this biased network. Our results are shown in Figure 7. Our method is clearly able to articulate that adding a square in the upper left makes something an 8, whereas the other methods are not able to explain this undesirable bias.

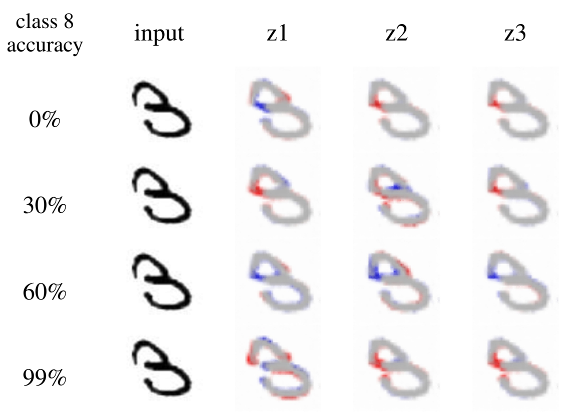


Figure 6: The explanations of "why not 8?" using the same latent space model for networks training at different accuracies. Right three columns are for different starting points of the optimization for $\mathbf{z}$. As the accuracy of the network for class "8" increases, the generated explanations are getting closer truth as well.

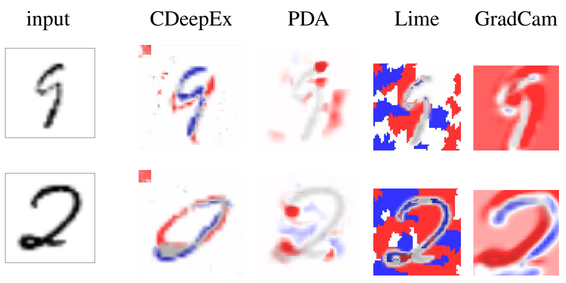


Figure 7: Explanations for "why not 8?" for a network trained on data in which every 8 has a small square in the upper-left.

4.4 CelebA Dataset

We ran experiments on the CelebA dataset [21]. We trained a Vgg16 network on a subset of the images with target labels of "blonde female," "dark-haired male," and "dark-haired female." (There were not sufficient numbers of blonde males in the training set.) The resulting classifier achieves 95% accuracy. For the generator network, we trained an auto encoder (AE) to generate images. The encoder consists of five convolutional layers with a kernel size of 3, a stride of 2, and a ReLU non-linearity. The number of filters for each layer are 16, 64, 128, 256, 512, and 1024.

We asked questions about why the network chose the hair color it did, and not the other hair color for the same gender. Our goal is to discover if the network learned the correct concept class. The right column of Figure 9 shows that xGEMs produces a diffuse explanation that covers most of the face. This is because it is missing the necessary constraints to keep the explanation just between the classes of blonde females and dark-haired females. It produces explanations that change the result to male (changing the facial structure), despite that the question does not ask about this class. The middle column shows that our method is more targeted to the real difference learned by the classifier: The difference learned by the classifier is in the eyes! This example shows that the model learned the wrong features compared with those we expected to associate with the label.

The top row of Figure 8 shows experiments with imposed changes

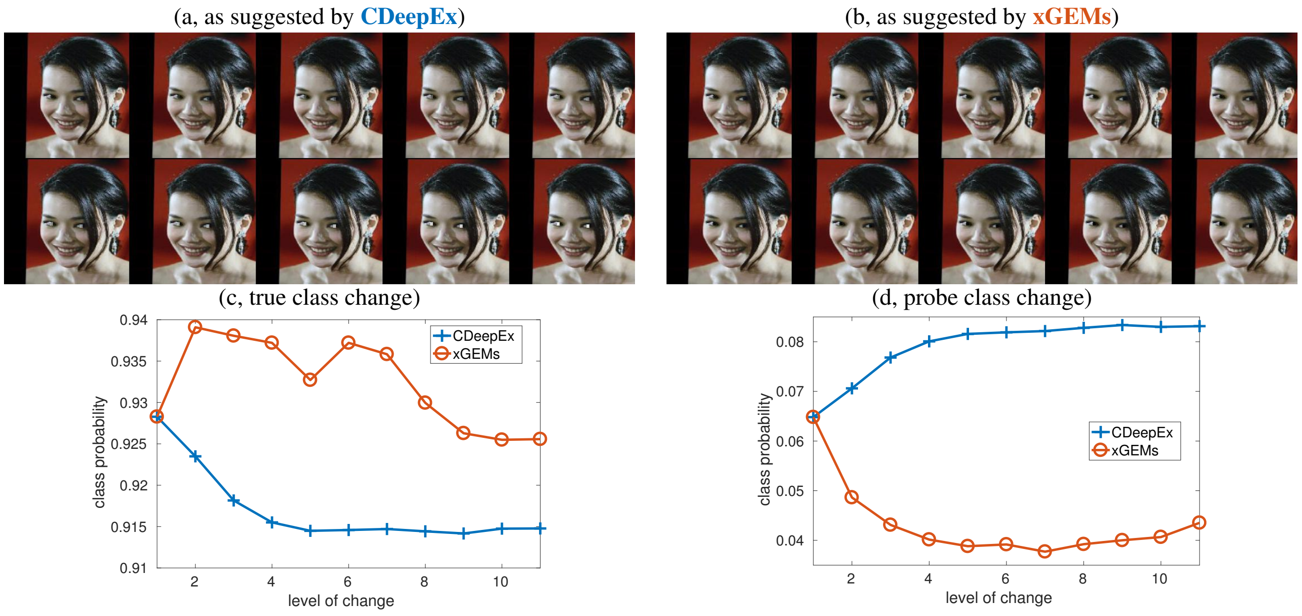


Figure 8: (a) we manually increase eye intensity from its original value to maximum, as suggested by our method, CDeepEx. (b) we decrease eye intensity from its original value to minimum, as suggested by xGEMs. (c) and (d) show the plots of how the discriminator network d's outputs change for the probe (blonde female) and reported/true (dark-haired female) classes. this demonstrates that xGEMs generates the wrong explanation of the correlation of the hair with the eyes, whereas the change suggested by CDeepEx correctly moves the reported class of the discriminator network from the true class (dark hair) to the probe class (blonde).

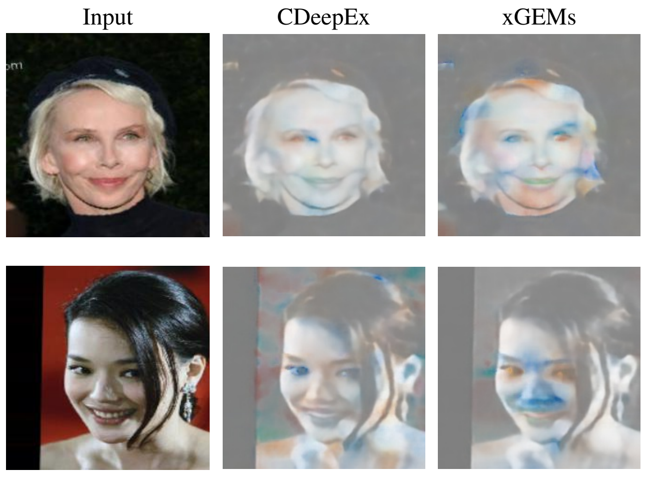


Figure 9: CDeepEX vs xGEMs experiments. The top row is the result of a query "why is the person not dark-haired?" on a blonde female. We see that the network pays more attention to the eye color than the hair color. In the second row, the probe class is blonde and true class is dark-haired, for the inverse query. We can see generated explanations from xGEMs and CDeepEx disagree over the "sign" of change in the eyes. See Figure 8 for more details.

that demonstrate that the learned discriminative network learned eye color and not hair color. In particular, lightening the eyes changes the reported hair color to blonde. However, xGEMs (which does not have the constraint on non-probe, non-true classes), gave the reverse explanation (darkening the eyes would produce a blonde classification) which was not borne out by our experiment.

Clearly, with better training or a more complete dataset, D could probably have learned the "correct" concept. The purpose of CDeepEx is not to find the "correct" differences in the classes, but rather the differences that the network D has decided on. Most critically, our method can identify when the classifier D has *not* properly generalized the training set. This CelebA result directly shows such an example, and our method is able to clearly explain what the classifier *did* detect (eye color). This information can be used to build (or erode) trust in the classifier and to suggest training or deployment changes that would help correct discovered errors.

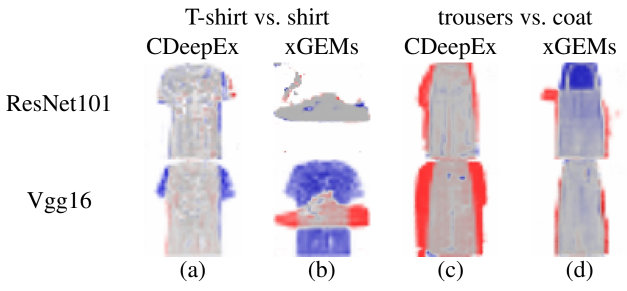


Figure 10: Top and bottom rows are the results for ResNet101 and VGG16 respectively. (a) Change from T-shirt to a Shirt with constraints. (b) T-shirt to shirt without constraints. (c) Change from trousers to coat with constraints. (d) Change from trousers to coat without constraints.

4.5 Fashion MNIST

We trained two different networks for D on the Fashion MNIST dataset: Vgg16 [32] and ResNet101 [10]. The testing accuracy for both networks is 92%. For the generating network, G, we used the structures and learning method presented by [1] with latent space of size 200. We then illustrate our method's ability to gain insight into the robustness of the classifiers through contrastive explanations. The generator network's structure is the same as the MNIST's generator network's structure.

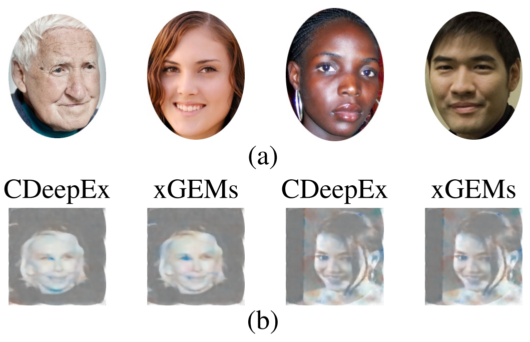


Figure 11: Cross-dataset comparison, latent-space model learned on a different dataset than the discriminative network. (a) Samples from latent-space training dataset. (b) Comparison between CDeepEx and xGEMs. See Figure 9.

Figure 10 shows the generated explanations with and without constraints. Comparing CDeepEx with xGEMs, we can see the importance of the constraints. Without them (xGEMs), the results refer to other irrelevant classes.

Using CDeepEx, it is clear that Vgg16 learned more general concepts than ResNet101. The first column shows that ResNet101 learned very subtle differences on the surface of the T-shirt to distinguish it from a (long-sleeved) shirt. By contrast, Vgg16 understands removing the short sleeves makes the appearance of a shirt with long sleeves. In the "trousers vs. coat" example, ResNet101 believes that adding a single sleeve will change the trouser into a coat, while Vgg16 requires both sleeves.

4.6 Training on Different Datasets

Finally, we show that the generator model can be trained on different dataset and still be robust for generating explanations. We train our AE generator model on Adult10Kfaces [12] dataset and test it on a VGG16 network learned on the CelebA dataset. Figure 11a shows some samples from Adult10Kfaces. Each picture is inside an oval which is faintly visible when generating explanations, confirming that the generator is trained on Adult10Kfaces dataset. Looking at the results for CDeepEx in Figure 11b shows that the explanations are consistent with those from Figure 9, in which the latent-space model was estimated from the CelebA dataset itself.

5 Conclusions

Our formulation draws on three ideas: The explanation should be in the space of natural inputs to aid communication with humans, should be an example that is maximally ambiguous between the true and probe classes, and should not be confused with other classes. The method does not require knowledge of or heuristics related to the architecture or modality of the network. The explanations can point to unintended correlations in the input data that are expressed in the resulting learned network.

Our contrastive explanation method (CDeepEx) provides an effective method for querying a network to discover its representations and biases. We demonstrated the quality of our method, compared to other current methods and illustrated how these contrastive explanations can shed light on the robustness of a learned network.

ACKNOWLEDGEMENTS

The authors thank the referees for their comments, which helped improve this paper. Feghahati and Pazzani were supported with funding from the DARPA Explainable AI Program under a contract from NRL. The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the DoD or the U.S. Government. Shelton was supported in part by the National Science Foundation (IIS 1510741).

REFERENCES

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou, 'Wasserstein generative adversarial networks', in *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, (2017).

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba, 'Network dissection: Quantifying interpretability of deep visual representations', in *Computer Vision and Pattern Recognition*, (2017).

[3] Dimitri P. Bertsekas, *Nonlinear Programming*, chapter 4, Athena Scientific, second edn., 1999.

[4] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud, 'Explaining image classifiers by counterfactual generation', in *International Conference on Learning Representations*, (2019).

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille, 'Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs', *IEEE Trans. Pattern Anal. Mach. Intell.*, **40**(4), 834–848, (2018).

[6] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu, 'Towards explanation of dnn-based prediction with guided feature inversion', in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*, KDD '18, pp. 1358–1367, New York, NY, USA, (2018). ACM.

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent, 'Visualizing higher-layer features of a deep network', Technical Report 1341, Département d'Informatique et Recherche Opérationnelle, Université de Montréal, (June 2009).

[8] Ruth Fong and Andrea Vedaldi, 'Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (June 2018).

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, 'Generative adversarial nets', in *Advances in Neural Information Processing Systems 27*, eds., Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, 2672–2680, Curran Associates, Inc., (2014).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778, (2016).

[11] Lisa Anne Hendricks, Zeynep Akata, Jeff Donahue, Bernt Schiele, and Trevor Darrell, 'Generating visual explanations', in *ECCV*, (2016).

[12] Phillip Isola, Devi Parikh, Antonio Torralba, and Aude Oliva, 'Understanding the intrinsic memorability of images', in *Advances in Neural Information Processing Systems 24*, eds., J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, 2429–2437, Curran Associates, Inc., (2011).

[13] Jason Jo and Yoshua Bengio, 'Measuring the tendency of cnns to learn surface statistical regularities', *CoRR*, **abs/1711.11561**, (2017).

[14] Shalmali Joshi, Oluwasanmi Koyejo, Been Kim, and Joydeep Ghosh, 'xGEMs: Generating examplars to explain black-box models', *ArXiv e-prints*, (June 2018).

[15] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Kristof T. Schütt, Maximilian Alber, Sven Dähne, Dumitru Erhan, and Been Kim. The (un)reliability of saliency methods, 2018.

[16] Diederik P. Kingma and Max Welling, 'Auto-encoding variational bayes', *CoRR*, **abs/1312.6114**, (2013).

[17] Pang Wei Koh and Percy Liang, 'Understanding black-box predictions via influence functions', in *ICML*, (2017).

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in Neural Information Processing Systems 25*, eds., F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, 1097–1105, Curran Associates, Inc., (2012).

[19] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki, 'The dangers of post-hoc interpretability: Unjustified counterfactual explanations', in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2801–2807. International Joint Conferences on Artificial Intelligence Organization, (7 2019).

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, **86**(11), 2278–2324, (1998).

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang, 'Deep learning face attributes in the wild', in *Proceedings of International Conference on Computer Vision (ICCV)*, (December 2015).

[22] Ari S Morcos, David GT Barrett, Matthew Botvinick, and Neil C Rabinowitz, 'On the importance of single directions for generalization', in *ICLR*, (2018).

[23] Jose Oramas, Kaili Wang, and Tinne Tuytelaars, 'Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks', in *International Conference on Learning Representations*, (2019).

[24] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrback, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach, 'Multimodal explanations: Justifying decisions and pointing to the evidence', in *CVPR*, (2018).

[25] Alec Radford, Luke Metz, and Soumith Chintala, 'Unsupervised representation learning with deep convolutional generative adversarial networks', *CoRR*, **abs/1511.06434**, (2015).

[26] Pranav Rajpurkar, Jeremy Irvin, Aarti Bagul, Daisy Ding, Tony Duan, Hershel Mehta, Brandon Yang, Kaylie Zhu, Dillon Laird, Robyn L. Ball, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Ng. Mura dataset: Towards radiologist-level abnormality detection in musculoskeletal radiographs, 2017. cite arxiv:1712.06957.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, '"Why should I trust you?": Explaining the predictions of any classifier', in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1135–1144, New York, NY, USA, (2016). ACM.

[28] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller, 'Evaluating the visualization of what a deep neural network has learned', *IEEE Transactions on Neural Networks and Learning Systems*, **28**(11), 2660–2673, (2017).

[29] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, 'Grad-cam: Visual explanations from deep networks via gradient-based localization', in *The IEEE International Conference on Computer Vision (ICCV)*, (Oct 2017).

[30] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje, 'Learning important features through propagating activation differences', *CoRR*, **abs/1704.02685**, (2017).

[31] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 'Deep inside convolutional networks: Visualising image classification models and saliency maps', *CoRR*, **abs/1312.6034**, (2013).

[32] Karen Simonyan and Andrew Zisserman, 'Very deep convolutional networks for large-scale image recognition', *CoRR*, **abs/1409.1556**, (2014).

[33] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller, 'Striving for simplicity: The all convolutional net', *ICLR*, **abs/1412.6806**, (2015).

[34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, 'Going deeper with convolutions', in *Computer Vision and Pattern Recognition (CVPR)*, (2015).

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf, 'Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms', *ArXiv e-prints*, (2017).

[36] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson, 'Understanding neural networks through deep visualization', *CoRR*, **abs/1506.06579**, (2015).

[37] Matthew D. Zeiler and Rob Fergus, 'Visualizing and understanding convolutional networks', in *ECCV*, (2014).

[38] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang, 'Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising', *IEEE Transactions on Image Processing*, **26**, (2017).

[39] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba, 'Learning Deep Features for Discriminative Localization.', *CVPR*, (2016).

[40] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling, 'Visualizing deep neural network decisions: Prediction difference analysis', *ICLR*, (2017).