Generating Contrastive Explanations with Monotonic Attribute Functions

Ronny Luss[1]*, Pin-Yu Chen[1], Amit Dhurandhar[1], Prasanna Sattigeri[1],
Karthikeyan Shanmugam[1], and Chun-Chen Tu[2]

May 31, 2019

Abstract

Explaining decisions of deep neural networks is a hot research topic with applications in medical imaging, video surveillance, and self driving cars. Many methods have been proposed in literature to explain these decisions by identifying relevance of different pixels. In this paper, we propose a method that can generate contrastive explanations for such data where we not only highlight aspects that are in themselves sufficient to justify the classification by the deep model, but also new aspects which if added will change the classification. One of our key contributions is how we define "addition" for such rich data in a formal yet humanly interpretable way that leads to meaningful results. This was one of the open questions laid out in [5], which proposed a general framework for creating (local) contrastive explanations for deep models. We showcase the efficacy of our approach on CelebA and Fashion-MNIST in creating intuitive explanations that are also quantitatively superior compared with other state-of-the-art interpretability methods.

1 Introduction

With the explosion of deep learning [7] and its huge impact on domains such as computer vision and speech, amongst others, many of these technologies are being implemented in systems that affect our daily lives. In many cases, a negative side effect of deploying these technologies has been their lack of transparency [25], which has raised concerns not just at an individual level [33] but also at an organization or government level [36].

There have been many methods proposed in literature [1, 14, 19, 23, 31] that explain predictions of deep neural networks based on the relevance of different features or pixels/superpixels for an image. Recently, an approach called contrastive explanations method (CEM) [5] was proposed which highlights

*First four authors have equal contribution. 1 and 2 indicate affiliations to IBM Research and University of Michigan respectively.

not just correlations or relevances but also features that are minimally sufficient to justify a classification, referred to as pertinent positives (PPs). CEM additionally outputs a minimal set of features, referred to as pertinent negatives (PNs), which when made non-zero or added, alter the classification and thus should remain absent in order for the original classification to prevail. For example, when justifying the classification of a handwritten image of a 3, the method will identify a subset of non-zero or on-pixels within the 3 which by themselves are sufficient for the image to be predicted as a 3 even if all other pixels are turned off (that is, made zero to match background). Moreover, it will identify a minimal set of off-pixels which if turned on (viz. a horizontal line of pixels at the right top making the 3 look like a 5) will alter the classification. Such forms of explanations are not only common in day-to-day social interactions (viz. the twin without the scar) but are also heavily used in fields such as medicine and criminology [5] with arguments for PNs being the most important aspect of an explanation [18].

To identify PNs, addition is easy to define for grayscale images where a pixel with a value of zero indicates no information and so increasing its value towards 1 indicates addition. However, for colored images with rich structure, it is not clear what is a “no information" value for a pixel and consequently what does one mean by addition. Defining addition in a naive way such as simply increasing the pixel or red-green-blue (RGB) channel intensities can lead to uninterpretable images as the relative structure may not be maintained with the added portion being not necessarily interpretable. Moreover, even for grayscaled images just increasing values of pixels may not lead to humanly interpretable images nor is there a guarantee that the added portion can be interpreted even if the overall image is realistic and lies on the data manifold.

In this paper, we overcome these limitations by defining “addition" in a novel way which leads to realistic images with the additions also being interpretable. To showcase the general applicability of our method to various settings, we first experiment with CelebA [17] where we apply our method to a data manifold learned using a generative adversarial network (GAN) [7] trained over the data and by building attribute classifiers for certain high-level concepts (viz. lipstick, hair color) in the dataset. We create realistic images with interpretable additions. Our second experiment is on Fashion-MNIST [35] where the data manifold is learned using a variational autoencoder (VAE) [15] and certain (interpretable) latent factors (as no attributes are available) are used to create realistic images with, again, interpretable additions. These two cases show that our method can be applied to colored as well as grayscale images and to datasets that may or may not have high level attributes.

2 Related Work

There have been many methods proposed in literature that aim to explain reasons for their decisions. These methods may be globally interpretable – rule/decision lists [32, 34], or exemplar based – prototype exploration [8, 12], or inspired by psychometrics [9] or interpretable generalizations of linear models [3]. Moreover, there are also works that try to formalize interpretability [6].

A survey by [19] mainly explores two methods for explaining decisions of neural networks: i) Prototype selection methods [21,22] that produce a prototype for a given class, ii) Local explanation methods that highlight relevant input features/pixels [1, 14, 25, 28]. Belonging to this second type there are multiple local explanation methods that generate explanations for images [23, 29, 31] and some others for NLP applications [16]. There are also works [13] that highlight higher level concepts present in images based on examples of the concept provided by the user. These methods mostly focus on features that are present, although they may highlight negatively contributing features to the final classification. In particular, they do not identify concepts or features that are minimally sufficient to justify the classification or those that should be necessarily absent to maintain the original classification. There are also evaluation methods that perturb the input and remove features [27] to verify their importance, but these methods can only evaluate an explanation and do not find one.

Recently, there have been works that look beyond relevance. In [26], the authors try to find features that, with almost certainty, indicate a particular class. These can be seen as global indicators for a particular class. Of course, these may not always exist for a dataset. There are also works [37] that try to find stable insight that can be conveyed to the user in a (asymmetric) binary setting for medium-sized neural networks. The most relevant work to our current endevour is [5] and, as mentioned before, it cannot be directly applied when it comes to explaining colored images or images with rich structure.

3 Methodology

We now describe the methodology for generating contrastive explanations for images. We first describe how to identify PNs, which involves the key contribution of defining “addition" for colored images in a meaningful way. We then describe how to identify PPs, which also utilizes this notion of adding attributes. Finally, we provide the algorithmic details for solving the optimization problems in Algorithm 1.

We first introduce some notation. Let $\mathcal{X}$ denote the feasible input space with $(\mathbf{x}_0, t_0)$ being an example such that $\mathbf{x}_0 \in \mathcal{X}$ and t_0 is the predicted label obtained from a neural network classifier. Let S denote the set of superpixels that partition $\mathbf{x}_0$ with $\mathcal{M}$ denoting a set of binary masks which when applied to $\mathbf{x}_0$ produce images $\mathcal{M}(x_0)$ by selecting the corresponding superpixels from S. Let M_x denote the mask corresponding to image $\mathbf{x} = M_x(\mathbf{x}_0)$.

If $\mathcal{D}(.)$ denotes the data manifold (based on GAN or VAE), then let z denote the latent representation with z_x denoting the latent representation corresponding to input $\mathbf{x}$ such that $\mathbf{x} = \mathcal{D}(z_x)$. Let k denote the number of (available or learned) interpretable features (latent or otherwise) which represent meaningful concepts (viz. moustache, glasses, smile) and let $g_i(.), \forall i \in \{1, ..., k\}$, be corresponding functions acting on these features with higher values indicating presence of a certain visual concept while lower values indicating its absense. For example, CelebA has different high-level (interpretable) features for each image such as whether the person has black hair or high cheekbones. In this case, we could build binary classifiers for each of the features where a 1 would indicate presence of black hair or high cheekbones, while a zero would mean its absense. These classifiers would be the $g_i(.)$ functions. On the other hand, for datasets with no high-level interpretable features, we could find latents by learning disentangled representations and choose those latents (with ranges) that are interpretable. Here the g_i functions would be an identity map or negative identity map depending on which direction adds a certain concept (viz. sleeveless shirt to a long sleeve one). We note that these attribute functions could be used as latent features for the generator in a causal graph (e.g., [20]), or given a causal graph for the desired attributes, we could learn these functions from the architecture in [20].

Our procedure for finding PNs and PPs involves solving an optimization problem over the variable $\boldsymbol{\delta}$ which is the outputted image. We denote the prediction of the model on the example $\mathbf{x}$ by $f(\mathbf{x})$, where $f(\cdot)$ is any function that outputs a vector of prediction scores for all classes, such as prediction probabilities and logits (unnormalized probabilities) that are widely used in neural networks.

Algorithm 1 Contrastive Explanations Method using Monotonic Attribute Functions (CEM-MAF)

```
Input: Example $(\mathbf{x}_0, t_0)$, latent representation of $\mathbf{x}_0$ denoted $z_{\mathbf{x}_0}$, neural network classification model $f(.)$, set of (binary) masks $\mathcal{M}$, and $k$ monotonic feature functions $g = \{g_1(.), ..., g_k(.)\}$.
1) Solve (1) on example $(\mathbf{x}_0, t_0)$ and obtain $\boldsymbol{\delta}^{\text{PN}}$ as the minimizing Pertinent Negative.
2) Solve (2) on example $(\mathbf{x}_0, t_0)$ and obtain $\boldsymbol{\delta}^{\text{PP}}$ as the minimizing Pertinent Positive.
return $\boldsymbol{\delta}^{\text{PN}}$ and $\boldsymbol{\delta}^{\text{PP}}$. {Code provided in supplement.}
```

3.1 Pertinent Negatives (PNs)

To find PNs, we want to create a (realistic) image that lies in a different class than the original image but where we can claim that we have (minimally) “added" things to the original image without deleting anything to obtain this new image. If we are able to do this, we can say that the things that were added, which we call PNs, should be necessarily absent from the original image in order for its classification to remain unchanged.

The question is how to define "addition" for colored or images with rich structure. In [5], the authors tested on grayscale images where intuitively it is easy to define addition as increasing the pixel values towards 1. This, however, does not generalize to images where there are multiple channels (viz. RGB) and inherent structure that leads to realistic images, and where simply moving away from a certain pixel value may lead to unrealistic images. Moreover, addition defined in this manner may be completely uninterpretable. This is true even for grayscale images where, while the final image may be realistic, the addition may not be. The other big issue is that for colored images the background may be any color which indicates no signal and object pixels which are lower in value in the original image if increased towards background will make the object imperceptible in the new image, although the claim would be that we have “added" something. Such counterintuitive cases arise for complex images if we maintain their definition of addition.

Given these issues, we define addition in a novel manner. To define addition we assume that we have high-level interpretable features available for the dataset. Multiple public datasets [17, 38] have high-level interpretable features, while for others such features can be learned using unsupervised methods such as disentangled representations [15] or supervised methods where one learns concepts through labeled examples [13]. Given k such features, we define functions $g_i(.), \forall i \in \{1, ..., k\}$, as before, where in each of these functions, increasing value indicates addition of a concept. Using these functions, we define addition as introducing more concepts into an image without deleting any existing concepts. Formally, this corresponds to never decreasing the $g_i(.)$ from their original values based on the input image, but rather increasing them. However, we also want a minimal number of additions for our explanation to be crisp and so we encourage as few $g_i()$ as possible to increase in value (within their allowed ranges) that will result in the final image being in a different class. We also want the final image to be realistic and that is why we learn a manifold $\mathcal{D}$ on which we perturb the image, as we want our final image to also lie on it after the necessary additions.

This gives rise to the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{\delta}\in\mathcal{X}}\ & \gamma\sum_i \max\{g_i(\mathbf{x}_0) - g_i(\mathcal{D}(z_{\boldsymbol{\delta}})), 0\} + \beta\|g\left(\mathcal{D}(z_{\boldsymbol{\delta}})\right)\|_1 \\ & - c\cdot\min\{\max_{i\neq t_0}[f(\boldsymbol{\delta})]_i - [f(\boldsymbol{\delta})]_{t_0}, \kappa\} + \eta\|\mathbf{x}_0 - \mathcal{D}(z_{\boldsymbol{\delta}})\|_2^2 + \nu\|z_{\mathbf{x}_0} - z_{\boldsymbol{\delta}}\|_2^2.\end{aligned} \tag{1}$$

The first two terms in the objective function here are the novelty for PNs. The first term encourages the addition of attributes where we want the $g_i(.)$s for the final image to be no less than their original values. The second term encourages minimal addition of interpretable attributes. The third term is the PN loss from [5] and encourages the modified example $\boldsymbol{\delta}$ to be predicted as a different class than $t_0 = \arg\max_i[f(\mathbf{x}_0)]_i$, where $[f(\boldsymbol{\delta})]_i$ is the i-th class prediction score of $\boldsymbol{\delta}$. The hinge-like loss function pushes the modified example $\boldsymbol{\delta}$ to lie in a different class than $\mathbf{x}_0$. The parameter $\kappa \geq 0$ is a confidence parameter that controls the separation between $[f(\boldsymbol{\delta})]_{t_0}$ and $\max_{i\neq t_0}[f(\boldsymbol{\delta})]_i$. The fourth ($\eta > 0$) and fifth terms ($\nu > 0$) encourage the final image to be close to the original image in the input and latent spaces. In practice, one could have a threshold for each of the $g_i(.)$, where only an increase in values only beyond that threshold would imply a meaningful addition. The advantage of defining addition in this manner is that not only are the final images interpretable, but so are the additions, and we can clearly elucidate which (concepts) should be necessarily absent to maintain the original classification.

3.2 Pertinent Positives (PPs)

To find PPs, we want to highlight a minimal set of important pixels or superpixels which by themselves are sufficient for the classifier to output the same class as the original example. More formally, for an example image $\mathbf{x}_0$, our goal is to find an image $\boldsymbol{\delta} \in \mathcal{M}(x_0)$ such that $\text{argmax}_i[\text{Pred}(\mathbf{x}_0)]_i = \text{argmax}_i[\text{Pred}(\boldsymbol{\delta})]_i$ (i.e. same prediction), with $\boldsymbol{\delta}$ containing as few superpixels and interpretable concepts from the original image as possible. This leads to the following optimization problem:

$$\min_{\boldsymbol{\delta} \in \mathcal{M}(\mathbf{x}_0)} \gamma \sum_i \max\{g_i(\boldsymbol{\delta}) - g_i(\mathbf{x}_0), 0\} - c \cdot \min\{[f(\boldsymbol{\delta})]_{t_0} - \max_{i \neq t_0}[f(\boldsymbol{\delta})]_i, \kappa\} + \beta \|M_{\boldsymbol{\delta}}\|_1. \tag{2}$$

The first term in the objective function here is the novelty for PPs and penalizes the addition of attributes since we seek a sparse explanation. The second term is the PP loss from [5] and is minimized when $[f(\boldsymbol{\delta})]_{t_0}$ is greater than $\max_{i \neq t_0}[f(\boldsymbol{\delta})]_i$ by at least $\kappa \geq 0$, which is a margin/confidence parameter. Parameters $\gamma, c, \beta \geq 0$ are the associated regularization coefficients.

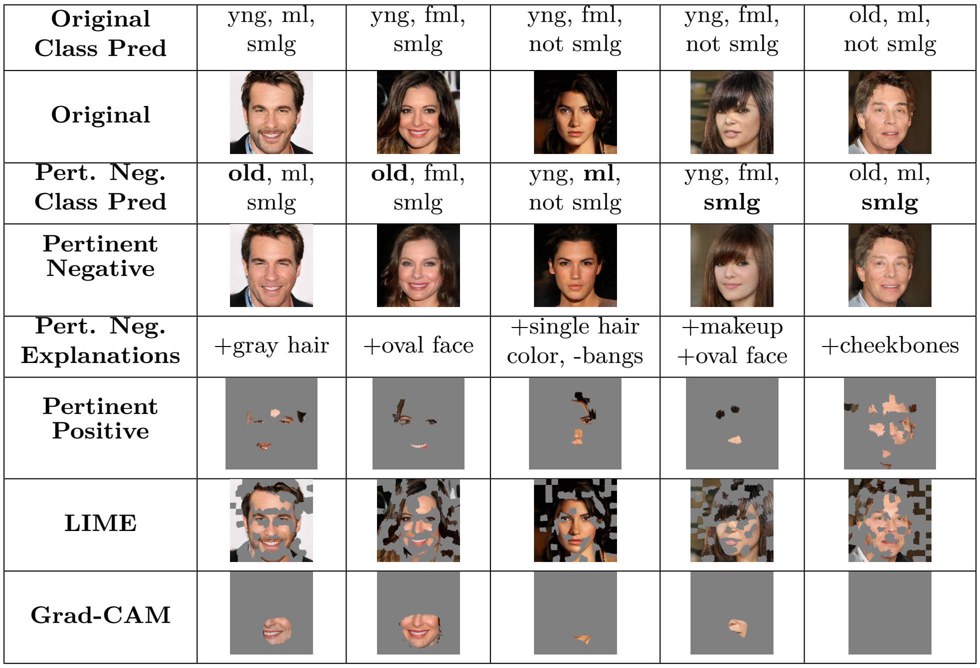
In the above formulation, we optimize over superpixels which of course subsumes the case of just using pixels. Superpixels have been used in prior works [25] to provide more interpretable results on image datasets and we allow for this more general option.

3.3 Optimization Details

To solve for PNs as formulated in (1), we note that the L_1 regularization term is penalizing a non-identity and complicated function $\|g(\mathcal{D}(z_{\boldsymbol{\delta}}))\|_1$ of the optimization variable $\boldsymbol{\delta}$ involving the data manifold $\mathcal{D}$, so proximal methods are not applicable. Instead, we use 1000 iterations of standard subgradient descent to solve (1). We find a PN by setting it to be the iterate having the smallest L_2 distance $\|z_{\mathbf{x}_0} - z_{\boldsymbol{\delta}}\|_2$ to the latent code $z_{\mathbf{x}_0}$ of $\mathbf{x}_0$, among all iterates where prediction score of solution $\boldsymbol{\delta}^*$ is at least $[f(\mathbf{x}_o)]_{t_0}$.

To solve for PPs as formulated in (2), we first relax the binary mask $M_{\boldsymbol{\delta}}$ on superpixels to be real-valued (each entry is between $[0,1]$) and then apply the standard iterative soft thresholding algorithm (ISTA) (see [2] for various references) that efficiently solves optimization problems with L_1 regularization. We run 100 iterations of ISTA in our experiments and obtain a solution $M_{\boldsymbol{\delta}^*}$ that has the smallest L_1 norm and satisfies the prediction of $\boldsymbol{\delta}^*$ being within margin κ of $[f(\mathbf{x}_o)]_{t_0}$. We then rank the entries in $M_{\boldsymbol{\delta}^*}$ according to their values in descending order and subsequently add ranked superpixels until the masked image predicts $[f(\mathbf{x}_o)]_{t_0}$.

A discussion of hyperparameter selection is held to the Supplement.



Original Class Pred	yng, ml, smlg	yng, fml, smlg	yng, fml, not smlg	yng, fml, not smlg	old, ml, not smlg
Original					
Pert. Neg. Class Pred	**old**, ml, smlg	**old**, fml, smlg	yng, **ml**, not smlg	yng, fml, **smlg**	old, ml, **smlg**
Pertinent Negative					
Pert. Neg. Explanations	+gray hair	+oval face	+single hair color, -bangs	+makeup +oval face	+cheekbones
Pertinent Positive					
LIME					
Grad-CAM					

Figure 1: CEM-MAF examples on CelebA using segmentation with 200 superpixels. Change from original class prediction is in bold in PN class prediction. Abbreviations in class predictions are as follows: yng for young, ml for male, fml for female, smlg for smiling.

4 Experiments

We next illustrate the usefulness of CEM-MAF on three image data sets - CelebA [17] and two subsets of Fashion-MNIST [35]. These datasets cover the gamut of color versus black/white images and having known high-level features versus derived disentangled features. CEM-MAF handles each scenario and offers explanations that are understandable by humans. These experiments exemplify the following observations:

- PNs offer intuitive explanations given a set of interpretable monotonic attribute functions. In fact, they seem to be the preferred form of explanation in many cases as describing a decision in isolation (viz. why is a shirt a shirt) using PPs, relevance, or heatmaps is not always informative.

- For colored images, PPs offer better direction as to what is important for the classification versus too much direction by LIME (shows too many features) or too little direction by Grad-CAM (only focuses on smiles), while for gray-scale images, neither PPs, LIME, or Grad-CAM are particularly informative versus PNs.

Table 1: Quantitative comparison of CEM-MAF, LIME, and Grad-CAM on the three datasets.

Dataset	**Method**	**# PP Feat**	**PP Acc**	**PP Corr**
CelebA	CEM-MAF	16.0	100	-.742
	LIME	75.1	30	.035
	Grad-CAM	17.6	30	.266
Fashion MNIST Clothes	CEM-MAF	62.7	100	-.766
	LIME	226.2	90	-.787
	Grad-CAM	344.0	100	.077
Fashion MNIST Shoes	CEM-MAF	15.7	100	-.947
	LIME	194.1	100	-.800
	Grad-CAM	202.3	100	-.500

- PPs and PNs offer the guarantee of being 100% accurate in maintaining or changing the class respectively as seen in Table 1 versus LIME or Grad-CAM.

- Both proximity in the input and latent space along with sparsity in the additions play an important role in generating good quality contrastive explanations.

4.1 CelebA with Available High-Level Features

CelebA is a large-scale dataset of celebrity faces annotated with 40 attributes [17].

4.1.1 Setup

The CelebA experiments explain an 8-class classifier learned from the following binary attributes: Young/Old, Male/Female, Smiling/Not Smiling. We train a Resnet50 [10] architecture to classify the original CelebA images. We selected the following 11 attributes as our $\{g_i\}$ based on previous studies [15] as well as based on what might be relevant for our class labels: High Cheekbones, Narrow Eyes, Oval Face, Bags Under Eyes, Heavy Makeup, Wearing Lipstick, Bangs, Gray Hair, Brown Hair, Black Hair and Blonde Hair. Note that this list does not include the attributes that define the classes because an explanation for someone that is smiling which simply says they are smiling would not be useful. Note that the usefulness of CEM-MAF is directly a function of the accuracies of the attribute functions. See Supplement for details on training these attribute functions and the GAN used for generation.

4.1.2 Observations

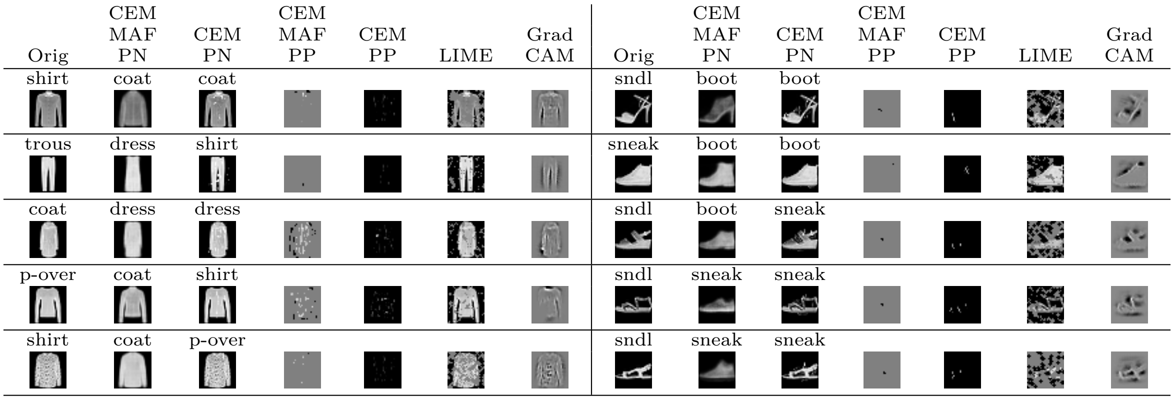
Results on five images are exhibited in Figure 1 using a segmentation into 200 superpixels[1]. The first two rows show the original class prediction followed by the original image. The next two rows show the pertinent negative's class prediction and the pertinent negative image. The fourth row lists the attributes that were modified in the original, i.e., the reasons why the original image is not classified as being in the class of the pertinent negative. The next row shows the pertinent positive image, which combined with the PN, gives the complete explanation. The final two rows illustrate different explanations that can be compared with the PP: one derived from locally interpretable model-agnostic explanations (LIME) [25] followed by a gradient-based localized explanation designed for CNN models (Grad-CAM) [24].

First consider class explanations given by the PPs. Age seems to be captured by patches of skin, sex by patches of hair, and smiling by the presence or absence of the mouth. One might consider the patches of skin to be used to explain young versus old. PPs capture a part of the smile for those smiling, while leave out the mouth for those not smiling. Visually, these explanations are simple (very few selected features) and quite useful although they require human analysis. In comparison, LIME selects many more features that are relevant to the prediction, and while also useful, requires even more human intervention to explain the classifier. Grad-CAM seems to always focus on the mouth (Grad-CAM is more useful for discriminative tasks) and does not always find a part of the image that is positively relevant to the prediction.

A performance comparison of PPs between CEM-MAF, LIME, and Grad-CAM is given in Table 1. Across colored images, CEM-MAF finds a much sparser subset of superpixels than LIME and is guaranteed to have the same prediction as the original image. Both LIME and Grad-CAM select features for visual explanations that often have the wrong prediction (low PP accuracy). A third measure, PP Correlation, measures the benefit of each additional feature by ranking the prediction scores after each feature is added (confidence in the prediction should increase) and correlating with the expected ranks (perfect correlation here would give -1). Order for LIME was determined by classifier weights while order for Grad-CAM was determined by colors in the corresponding heatmaps. CEM-MAF is in general best at selecting features that increase confidence.

More intuitive explanations are offered by the pertinent negatives in Figure 1. The first PN changes a young, smiling, male into an old, smiling, male by adding gray hair, so we can explain young as not having gray hair. While the gray hair is not visually apparent, the classifier has picked up on it. We also note that while his facial hair was removed by the PN, this cannot be part of our explanation because facial hair is not one of our attributes. Another way to explain being young, in the second column, is the absence of an oval face. The third PN changes a female into a male, and the female is explained by the absence of a single hair color (in fact, she has black hair with brown highlights)

[1] More examples in supplement.



(a) Fashion-MNIST clothes dataset

(b) Fashion-MNIST shoes dataset

Figure 2: CEM-MAF examples on Fashion-MNIST (a) clothes and (b) shoes dataset. CEM-MAF PN/PP are compared with CEM PN/PP from [5], LIME, and Grad-CAM. Note the abbreviations: p-over for pullover, trous for trousers, boot for ankleboot, and sneak for sneakers.

and the presence of bangs. While the presence of bangs is intuitive, it is selected because our constraints of adding features to form PNs can be violated due to enforcing the constraints with regularization. The last two column explain a straight face (not smiling), which is given by the absence of high cheekbones or the absense of an oval face (since your face can become more oval when you raise your cheekbones).

4.2 Fashion-MNIST with Learned Disentangled Features

Fashion-MNIST is a large-scale image dataset of various fashion items (e.g., coats, trousers, sandals).

4.2.1 Setup

Two datasets are created as subsets of Fashion-MNIST, one for clothes (teeshirts, trousers, pullovers, dresses, coats, and shirts) and one for shoes (sandals, sneakers, and ankleboots). As with CelebA, we need to generate new images for PNs that are realistic, and thus, a VAE was trained to learn the Fashion-MNIST manifold. However, this data does not have annotated features as in CelebA, so we learn the features using disentanglement following a recent variant of VAE called DIP-VAE [15] (see Supplement for more details).

We can then use these disentangled latent codes/features in lieu of the ground truth attributes. Based on what might be relevant to the clothes and shoes classes, we use four dimensions from the latent code as the attributes, corresponding to sleeve length, shoe mass, heel length and waist size. Given these attributes, we learn two classifiers, one with six classes (clothes) and one with

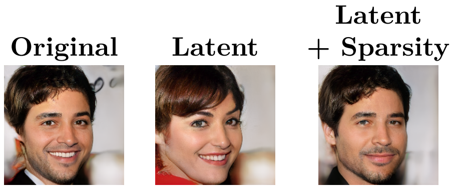


Figure 3: Illustration of different regularizations to obtain PN. Regularizing only proximity of latent representation explains why the male is not female, while also regularizing attribute sparsity explains why the male is smiling.

three classes (shoes). See the Supplement for a visualization of the attributes and details about the classifiers.

4.2.2 Observations

Results on five clothes images and five shoe images are shown in Figure 2 (a) and (b), respectively. In order to make a fair comparison with CEM, we do not segment the images but do feature selection pixel-wise. Note that CEM does not do pixel selection for PPs but rather modifies pixel values.

Let us first look at the PPs. They are mostly sparse explanations and do not give any visually intuitive explanations. Both LIME and Grad-CAM, by selecting many more relevant pixels, offers a much more visually appealing explanation. However, these explanations simply imply, for example, that the shirt in the first row of Figure 2 (a) is a shirt because it looks like a shirt. These two datasets (clothes and shoes) are examples where the present features do not offer an intuitive explanation. Table 1 again shows that CEM-MAF selects much fewer features, but here LIME does better at selecting useful features (PP Correlation close to -1 for clothes). Additionally, LIME and Grad-CAM both have high PP Accuracy, but that is due to selecting many more features.

Rather, constrastive explanations (relative to other items) about what is absent leads to the most intuitive explanations for these images. That same shirt is a shirt because it does not have wide sleeves and is not wide around the waist. In the second row of Figure 2 (b) are classified as trousers because the connection between the legs is absent, and in the fifth row, the item is classified as a shirt because a solid color (more likely on a coat) is absent (i.e., there is a pattern). In the first row of Figure 2 (b), the item is a sandal because it is not thick (i.e., it is an open shoe), while the second item is a sneaker (rather than an ankleboot) because it is missing a heal. The other rows in Figure 2 demonstrate similar explanations.

4.3 Value of L_1 Regularization in PNs

Two key regularizations in the pertinent negative optimization problem (1) maintain that the latent representations of the original image and its PN remain close and that only a few of the attribute functions exhibit changes . One might ask whether both regularizations are necessary, and in fact, the framework does allow for including either regularization separately or jointly by setting appropriate penalty values.

Figure 3 illustrates the usefulness of these regularizations on an image of a young, smiling, male (left image). Regularizing only the latent representation proximity results in a PN that is a young, smiling, female (middle image), from which we can explain that the original image is a male because of the absence of makeup, bangs, and brown hair. The brown hair makes sense because the original image is not defined by any hair color, while female hair color is often easier to detect because females color their hair more often resulting in stronger color [4, 30]. Interestingly, three attributes were used to explain the female PN. However, adding sparsity to the number of selected attributes results in the not smiling PN (right image) obtained by solely modifying the cheekbone attribute. The lesson is that both forms of regularization add something to the explanation. Proximity of latent representations keeps the image visually similar (a close inspection shows the male and female having similar facial structure, smile, nose, etc.), while sparsity of modifications can be used to keep the explanation simpler resulting in minimal additions.

5 Discussion

In the previous sections, we produced contrastive explanations by learning a data manifold. It is important to note that we do not necessarily need such a global data manifold to create our explanations, but rather a good *local* data representation around the image in question should be sufficient. Thus, for datasets where building an accurate global representation may be hard, if one can build a good local representation (viz. using conditional GANs), then it should still be possible to generate high quality contrastive explanations.

In this paper, we leveraged high level features that were readily available (viz. CelebA) as well as used generated ones (viz. Fashion-MNIST) based on unsupervised methods to produce contrastive explanations. As mentioned before one could also learn interpretable features in a supervised manner as done in [13] which we could also use to generate such explanations.

In summary, we have proposed a method to create contrastive explanations for image data with rich structure. The key novelty over previous work has been the way in which we define addition, which leads to realistic images and where the added information is also easy to interpret (viz. added makeup). Our results also showcase that using pertinent negatives might be the preferred form of explanation where it is not clear why a certain entity is what it is in isolation (e.g. based on relevance or pertinent positives), but could be explained much more crisply by contrasting it to another entity that closely resembles it (viz. adding a heal to a sneaker makes it look like an ankleboot).

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[3] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1721–1730, New York, NY, USA, 2015. ACM.

[4] Kavita Daswani. More men coloring their hair. *LA Times*, 2012.

[5] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, 2018.

[6] Amit Dhurandhar, Vijay Iyengar, Ronny Luss, and Karthikeyan Shanmugam. Tip: Typifying the interpretability of procedures. *arXiv preprint arXiv:1706.02952*, 2017.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

[8] Karthik Gurumoorthy, Amit Dhurandhar, and Guillermo Cecchi. Protodash: Fast interpretable prototype selection. *arXiv preprint arXiv:1707.01212*, 2017.

[9] Tsuyoshi Idé and Amit Dhurandhar. Supervised item response models for informative prediction. *Knowl. Inf. Syst.*, 51(1):235–257, April 2017.

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

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *ICLR*, 2018.

[12] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *In Advances of Neural Inf. Proc. Systems*, 2016.

[13] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors. *Intl. Conf. on Machine Learning*, 2018.

[14] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. In *Intl. Conference on Learning Representations (ICLR)*, 2018.

[15] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *Intl. Conf. on Learning Representations*, 2017.

[16] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

[17] 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.

[18] Tim Miller. Contrastive explanation: A structural-model approach. *CoRR*, abs/1811.03163, 2018.

[19] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 2017.

[20] Alexandros G. Dimakis Murat Kocaoglum Christopher Snyder and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. In *International Conference on Learning Representations (ICLR 2018)*, 2018.

[21] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016.

[22] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[23] Jose Oramas, Kaili Wang, and Tinne Tuytelaars. Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks. In *arXiv:1712.06302*, 2017.

[24] Abhishek Das Ramakrishna Vedantam Devi Parikh Ramprasaath R. Selvaraju, Michael Cogswell and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of IEEE Conference on Computer Vision (ICCV)*, October 2017.

[25] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, 2016.

[26] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[27] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. In *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[28] Su-In Lee Scott Lundberg. Unified framework for interpretable methods. In *In Advances of Neural Inf. Proc. Systems*, 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. *See https://arxiv.org/abs/1610.02391 v3*, 2016.

[30] SFR. 75% of women now color their hair compared to 7% in 1950. *South Florida Reporter*, 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] Guolong Su, Dennis Wei, Kush Varshney, and Dmitry Malioutov. Interpretable two-level boolean rule learning for classification. In *https://arxiv.org/abs/1606.05798*, 2016.

[33] Kush Varshney. Engineering safety in machine learning. In *https://arxiv.org/abs/1601.04126*, 2016.

[34] Fulton Wang and Cynthia Rudin. Falling rule lists. In *In AISTATS*, 2015.

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[36] Philip N. Yannella and Odia Kagan. Analysis: Article 29 working party guidelines on automated decision making under gdpr. 2018. https://www.cyberadviserblog.com/2018/01/analysis-article-29-working-party-guidelines-on-automated-decision-making-under-gdpr/.

[37] Xin Zhang, Armando Solar-Lezama, and Rishabh Singh. Interpreting neural network judgments via minimal, stable, and symbolic corrections. 2018. https://arxiv.org/abs/1802.07384.

[38] Shi Qiu Xiaogang Wang Ziwei Liu, Ping Luo and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Supplement

Note all model training for experiment is done using Tensorflow and Keras.

1 Hyperparameter selection for PN and PP

Find PNs is done by solving (1) which has hyperparameters $\kappa, \gamma, \beta, \eta, \mu, c$. The confidence parameter κ is the user's choice. We experimented with $\kappa \in \{0.5, 5.0\}$ and report results with $\kappa = 5.0$. We experimented with $\gamma \in \{1, 100\}$ and report results with $\gamma = 100$ which better enforces the constraint of only adding attributes to a PN. The three hyperparameters $\beta = 100, \eta = 1.0, \nu = 1.0$ were fixed. Sufficient sparsity in attributes was obtained with this value of β but further experiments increasing β could be done to allow for more attributes if desired. The results with selected η and ν were deemed realistic there so there was no need to further tune them. Note that Section 4.3 required experimenting with $\beta = 0$ to remove the attribute sparsity regularization. The last hyperparameter c was selected via the following search: Start with $c = 1.0$ and multiply c by 10 if no PN found after 1000 iterations of subgradient descent, and divided by 2 if PN found. Then run the next 1000 iterations and update c again. This search on c was repeated 9 times, meaning a total of $1000 \times 9 = 9000$ iterations of subgradient descent were run with all other hyperparameters fixed.

Find PPs is done by solving (2) which has hyperparameters κ, γ, β, c. Again, we experimented with $\kappa \in \{0.5, 5.0\}$ and report results with $\kappa = 5.0$. We experimented with $\gamma \in \{1, 100\}$ and report results with $\gamma = 100$ for the same reason as for PNs. The hyperparameter $\beta = 0.1$ because $\beta = 100$ as above was too strong and did not find PPs with such high sparsity (usually allowing no selection). The same search on c as described for PNs above was done for PPs, except this means a total of 9×100 iterations of ISTA were run with all other hyperparameters fixed to learn a single PP.

2 Additional CelebA Information

We here discuss how attribute classifiers were trained for CelebA, describe the GAN used for generation, and provide additional examples of CEM-MAF. CelebA datasets are available at `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`.

2.1 Training attribute classifiers for CelebA

For each of the 11 selected binary attributes, a CNN with four convolutional layers followed by a single dense layer was trained on 10000 CelebA images with Tensorflow's SGD optimizer with Nesterov using learning rate=0.001, decay=1e-6, and momentum=0.9. for 250 epochs. Accuracies of each classifiers are given in Table 2.

Table 2: CelebA binary attribute classifer accuracies on 1000 test images

Attribute	**Accuracy**
High Cheekbones	84.5%
Narrow Eyes	78.8%
Oval Face	63.7%
Bags Under Eyes	79.3%
Heavy Makeup	87.9%
Wearing Lipstick	90.6%
Bangs	93.0%
Gray Hair	93.5%
Brown Hair	76.3%
Black Hair	86.5%
Blonde Hair	93.1%

2.2 GAN Information

Our setup processes each face as a 224×224 pixel colored image. A GAN was trained over the CelebA dataset in order to generate new images that lie in the same distribution of images as CelebA. Specifically, we use the pretrained progressive GAN[2] in [11] to approximate the data manifold of the CelebA dataset. The progressive training technique is able to grow both the generator and discriminator from low to high resolution, generating realistic human face images at different resolutions.

2.3 Additional CelebA Examples

Figure 4 gives additional examples of applying CEM-MAF to CelebA. Similar patterns can be seen with the PPs. CEM-MAF provides sparse explanations highlighting a few features, LIME provides explanations (positively relevant superpixels) that cover most of the image, and Grad-CAM focuses on the mouth area.

3 Additional Fashion-MNIST Information

We here discuss how disentangled features were learned for Fashion-MNIST, how classifiers were trained, and provide additional examples of CEM-MAF. Fashion-MNIST datasets are available at `https://github.com/zalandoresearch/fashion-mnist`.

[2] `https://github.com/tkarras/progressive_growing_of_gans`

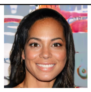
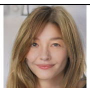
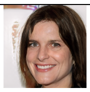
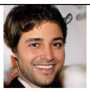
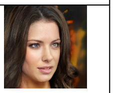
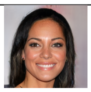
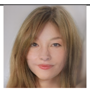
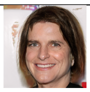
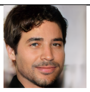
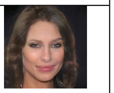
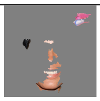
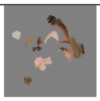
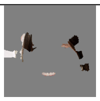
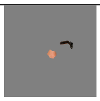
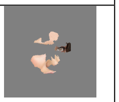
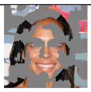
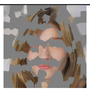
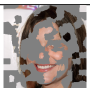
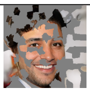
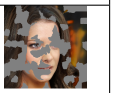
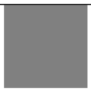
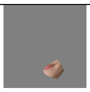
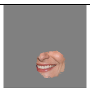
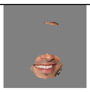
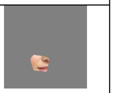
Original Class Pred	old, fml, not smlg	yng, fml, smlg	yng, fml, smlg	yng, ml, smlg	yng, fml, not smlg
Original					
Pert. Neg. Class Pred	**yng**, fml, smlg	yng, fml, **not smlg**	yng, **ml**, smlg	yng, ml, **not smlg**	**old**, fml, smlg
Pertinent Negative					
Pertinent Positive					
LIME					
Grad-CAM					

Figure 4: Additional CEM-MAF examples on CelebA using segmentation with 200 superpixels. Change from original class prediction is in bold in PN class prediction. Abbreviations in class predictions are as follows: yng for young, ml for male, fml for female, smlg for smiling.

3.1 Learning Disentangled Features for Fashion-MNIST

Our setup processes each item as a 28×28 pixel grayscale image. As with many real-world scenarios, Fashion-MNIST samples come without any supervision about the generative factors or attributes. For such data, we can rely on latent generative models such as VAE that aim to maximize the likelihood of generating new examples that match the observed data. VAE models have a natural inference mechanism baked in and thus allow principled enhancement in the learning objective to encourage disentanglement in the latent space. For inferring disentangled factors, inferred prior or expected variational posterior should be factorizable along its dimensions. We use a recent variant of VAE called DIP-VAE [15] that encourages disentanglement by explicitly matching the inferred aggregated posterior to the prior distribution. This is done by matching the covariance of the two distributions which amounts to decorrelating the dimensions of the inferred prior. Table 3 details the architecture for training DIP-VAE.

Figure 5 shows the disentanglement in these latent features by visualizing the VAE decoder's output for single latent traversals (varying a single latent between $[-3, 3]$ while keeping others fixed). For example, we can see that increasing the value of the second dimension z_2 of the latent code corresponds to increasing sleeve length while increasing the value of the third dimension z_3 corresponds to

Table 3: Details of the model architecture used for training DIP-VAE [15] on Fashion-MNIST.

	Architecture
Input	784 (flattened 28x28x1)
Encoder	FC 1200, 1200. ReLU activation.
Latents	16
Decoder	FC 1200, 1200, 1200, 784. ReLU activation.
Optimizer	Adam (lr = 1e-4) with mse loss

adding more material on the shoe.

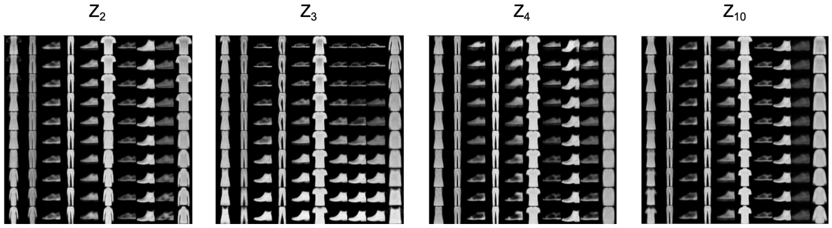


Figure 5: Qualitative results for disentanglement in Fashion MNIST dataset. The figure shows the DIP-VAE decoder's output for single latent traversals (varying a single latent between $[-3, 3]$ while keeping others fixed). The title of each image grid denotes the dimension of the latent code that was varied.

3.2 Training Fashion-MNIST classifiers

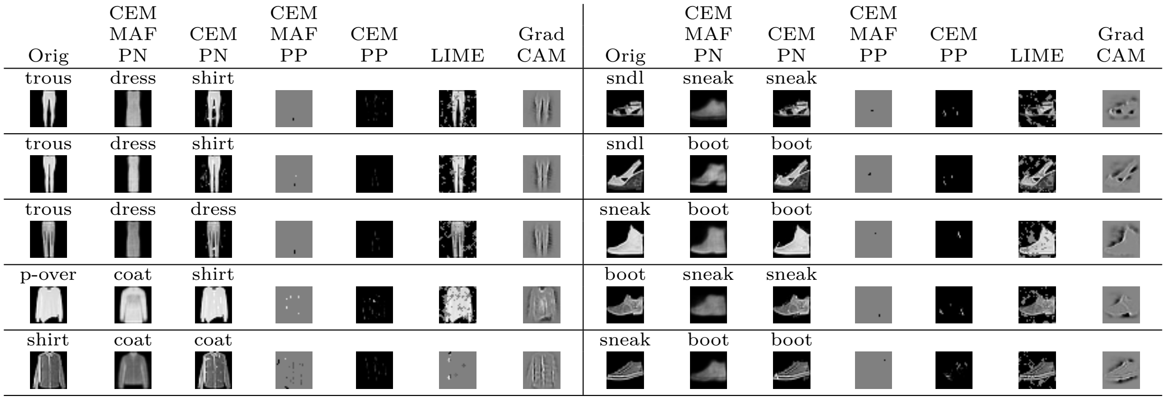
Two datasets are created as subsets of Fashion-MNIST, one for clothes (tee-shirts, trousers, pullovers, dresses, coats, and shirts) and one for shoes (sandals, sneakers, and ankleboots). We train CNN models for each of these subsets with two convolutional layers followed by two dense layer to classify corresponding images from original Fashion-MNIST dataset. See Table 4 for training details.

3.3 Additional Fashion-MNIST Examples

Figure 6 gives additional examples of applying CEM-MAF to Fashion-MNIST. Here, the PPs are not as useful as for CelebA because they are often too sparse. This could be alleviated by requiring more confidence (lower κ in CEM-MAF). Both LIME and Grad-CAM highlight most of the images which is also not particularly useful for explaining. This is a dataset where the PNs offer the most intuitive explanations.

Table 4: Details of the model architecture used for training classifiers on Fashion-MNIST.

	Architecture
Input	28x28x1
Shoes Classifier	Conv (5,5,32), MaxPool(2,2), Conv(5,5,64), MaxPool(2,2), Flatten, FC 1024, Dropout (rate=0.4), FC 3, Relu activation.
Clothes Classifier	Conv (5,5,32), MaxPool(2,2), Conv(5,5,64), MaxPool(2,2), Flatten, FC 1024, Dropout (rate=0.4), FC 6, Relu activation.
Optimizer	SGD (lr = 1e-3) with cross entropy loss



(a) Fashion-MNIST clothes dataset

(b) Fashion-MNIST shoes dataset

Figure 6: Additional CEM-MAF examples on Fashion-MNIST (a) clothes and (b) shoes dataset. CEM-MAF PN/PP use our new methods, and are compared with CEM PN/PP from [5], LIME, and Grad-CAM. Class labels are given above original and PN images. Note the abbreviations: p-over for pullover, trous for trousers, boot for ankleboot, and sneak for sneakers.