Learning Analogies between Classes to Create Counterfactual Explanations

Xiaomeng Ye[1,*], David Leake[2], Yu Wang[2], Ziwei Zhao[2] and David Crandall[2]

[1]*Berry College, Mount Berry GA 30149, USA*
[2]*Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington IN 47408, USA*

Abstract
Machine learning (ML) classification algorithms predict based on learned patterns of class characteristics. The patterns between different classes are less studied. Such patterns can be considered as analogies in classification domain: If a and c are from a class C_1, b and d are from a class C_2 and the two classes C_1 and C_2 are consistently similar in some features and different in some other features, then the analogy $a : b :: c : d$ holds, because the similarities (resp. differences) between a and b are equivalent to those between c and d. A class-to-class siamese network (C2C-SN) is a classifier neural network that trains on sample pairs from two classes C_1 and C_2, learns an inter-class pattern—i.e., an analogy—between the two classes and then can decides whether any two samples a and b holds in the analogy $a : b :: c_1 : c_2$ where c_1 and c_2 are representative samples from C_1 and C_2 respectively. If the analogy does exist, from sufficient random sample pairs the C2C-SN can learn without human knowledge or intervention. This paper proposes a general method to regularize any feature extraction ML model in a classification domain, by combining the ML model with a C2C-SN, to regulate the ML model to extract features important to inter-class analogies. It demonstrates an application of this method to learning and applying these analogies for counterfactual explanation in an image domain.

Keywords
Analogy, Case Adaptation, Case-based Reasoning, Counterfactual, Representation Learning

1. Introduction

Analogical reasoning research has been extensively studied for symbolic representations (e.g., [1, 2]), but is less commonly applied to non-symbolic data such as images until recent [3, 4]. Machine learning techniques have achieved great success for classifying complex data such as large language corpora, images, and audio data. However, models like neural networks generally learn the pattern from input to output while ignoring analogies between the categories. This paper presents research on learning analogies directly from non-symbolic data using machine learning techniques. Specifically, it focuses on analogy in an image classification domain, assuming that a similarity and difference pattern exists between a pair of classes C_1 and C_2. For x_1, x_2 of class C_1 and y_1, y_2 of class C_2 and x_1 is similar to and different from y_1 the same

IARML@IJCAI'2024: Workshop on the Interactions between Analogical Reasoning and Machine Learning, at IJCAI'2024, August, 2024, Jeju, South Korea
*Corresponding author.

xye@berry.edu (X. Ye); leake@iu.edu (D. Leake); yw173@iu.edu (Y. Wang); zz47@iu.edu (Z. Zhao); djcran@iu.edu (D. Crandall)
0000-0002-2289-1022 (X. Ye); 0000-0002-8666-3416 (D. Leake)

way as x_2 is for y_2, we consider $x_1 : y_1 :: x_2 : y_2$ as an analogy. The paper describes a general method that can be used to guide any machine learning (ML) model trainable with gradient descent, to extract features with analogy in mind.

We illustrate the method by applying it to use the analogies for counterfactual explanation [5]. Counterfactual explanations explain classifications by showing the effects of feature changes on the classification. By doing so, they both clarify the scope of the concepts and potentially provide actionable information for changing the outcome. For example, a counterfactual explanation for denying a loan might be that the applicant would have been approved if the applicant had had 5000 euro more savings. Given a domain described by symbolic features, counterfactual explanations can be generated by selectively changing feature values to generate examples across a decision boundary. For explaining classifications of image data, predefined feature sets are not available. Consequently, research on counterfactual explanations of image classifications has used ML techniques to extract and modify features [6, 7].

This paper demonstrates the usefulness of the learned analogies in counterfactual explanation in two image domain tasks. The method regularizes the feature extraction process by pairing it with a class-to-class siamese network (C2C-SN) and tuning it to extract features that are significant in inter-class analogies. A C2C-SN can rank features based on their importance in inter-class analogies. We hypothesize that modifying such features, which are most important for distinguishing classes, leads to good counterfactual explanations.

The paper begins with related work. It then discusses the process of training an upstream model with a C2C-SN to learn inter-class analogies and to rank features by their importance in these analogies. It presents three experiments testing the method. The first experiment combines the C2C-SN with a Variational Autoencoder (VAE) [8] and applies the resulted model to two data sets, MNIST [9] and FashionMNIST [10], showing that modifying single features top-ranked by C2C-SN feature importance often directly leads to desired class change. A second experiment combines C2C-SN with the PIECE algorithm [6], which can modify features exceptional for a class into their expected values, on the MNIST data set. A C2C-SN network can rank features and suggest which features to change while PIECE can carry out the change to generate a counterfactual. The counterfactuals generated are different but comparable in quality to previous results, supporting that C2C-SN can work with different upstream ML models and can guide a counterfactual explanation method with a new perspective. A third experiment combines C2C-SN with a VAE on the cars3d data set [11], revealing limitations of the approach and a potential direction for future research.

2. Background

2.1. Analogy Between Classes and Class-to-Class (C2C) Methodology

Machine learning classification methods generally classify based on similarity between members of a class. Less studied are the patterns of similarities and differences between classes. The pattern from one class to another is essentially an analogy between the two classes. For example, given two images of the digit 1, in roman and italic fonts ("1" and "*1*"), and two digit 7s ("7" and "*7*"), there exists the analogy that "1" is to "7" as "*1*" is to "*7*", matching the pattern of the analogical proportion "A is to B as C is to D". Moreover, such an analogy applies to many pairs of samples from the two classes: Here “1”s and “7”s share a vertical line, but “7”s always have a horizontal line on the top while “1”s do not. In other words, if 1s (or 7s) are similar to other 1s (or 7s) in a consistent way, then 1s and 7s are similar or different in a consistent way.

Inspired by siamese networks [12], which learn a similarity pattern between samples sharing the same class label, Ye et al. [13] developed class-to-class siamese networks (C2C-SNs) which learn the inter-class patterns where samples of one class are consistently different or similar on certain features when compared to samples of another class. Following the original paper, the inter-class pattern, or analogy, from a class C_1 to a class C_2 will be denoted as C_1-C_2. This paper modifies the original design slightly: Here a C2C-SN is a neural network that uses a shared network to extract features from two samples, calculates a feature difference $f_\triangle$ and a class label difference $C_\triangle$, and outputs a value p (between 0 and 1) indicating whether the feature difference $f_\triangle$ matches the analogy between the two classes indicated in $C_\triangle$.

This paper presents a general usage of C2C-SNs to highlight features relevant to inter-class analogies learned from data. By pairing a C2C-SN with any feature extraction model and training both together, the C2C-SN can (1) regularize the training of the feature extractor so the features extracted are useful in identifying inter-class analogies and (2) rank features based on their importance in a specific analogy.

2.2. Machine Learning Models that Learn Analogies

As discussed in the survey by Bounhas et al., [14], the analogy community has studied analogies for the purposes of classification (e.g., [15, 16, 17]) and explanation [18]. Analogical models are mainly either logical, algebraic, or complexity-based [19]. This paper takes a different path by using a neural model to learn the inter-class analogies directly from image data. The resulting model does not require the process of choosing a triplet a, b, c to classify (or explain) a sample d based on an analogy $a : b :: c : d$. Instead, a C2C-SN simply decides whether a pair $a : b$ match a pattern C_1-C_2.

Early analogy research explored concept analogies on small symbolic concepts [20], while more recent analogy research has applied modern ML techniques to domains such as tabular data [5], images [21] and natural language processing tasks [22, 23]. A comprehensive survey can be found in [24]. This paper presents a neural network method that can help any feature extraction model to find analogy-relevant features in classification domains, and demonstrates how the identification of such features can be used in counterfactual explanation in images. Lim [18] also identified the usage of analogy in factual and counterfactual explanations, but they experimented on Boolean data sets; Our work is in spirit aligned with Hüllermeier [5], who demonstrated analogy-based explanation in machine learning on a tabular data and suggested that analogy-based explanations can complement similarity-based explanations.

2.3. Counterfactual Explanation for Images

Counterfactual explanation in image domains has received much attention [7]. Generating counterfactual images often requires applying machine learning techniques to extract features, which are then modified for counterfactual explanation. For example, a VAE can encode sample inputs to embeddings that fall within a predetermined distribution (normally Gaussian). REVISE creates counterfactual by modifying latent features extracted by a VAE [25]. Theobald et al. [26] shows that REVISE can generate unrealistic images and improves upon it with *Clarity*.

Kenny and Keane [6] propose the PlausIble Exceptionality-based Contrastive Explanations (PIECE) algorithm, which "identifies probabilistically-low feature-values in the test image (i.e., exceptional features) and modifies them to be their expected values in the counterfactual class (i.e., normal features)". Feature importance methods measure how influential a feature is in a model's output. Two popular examples are LIME [27] and SHAP [28]. A recent survey on feature importance methods [29] links feature importance to counterfactual explanations, as they can both serve as local explanations for a model's behavior.

Our approach uses the Fisher et al. [30] model reliance method (as explained in Molnar [31]) to rank feature importance in inter-class analogies. This ranking guides a counterfactual explanation method to adapt a sample from a source class C_1 into a target class C_2 by prioritizing the modification of features important in the pattern C_1-C_2. It can be considered as an experience-guided counterfactual explanation method.

3. Proposed Method

Our method for counterfactual generation begins by extracting features regularized by a C2C-SN, then identifies the most important features for analogies, and generates counterfactuals by varying those features. This approach is generally applicable to any feature extractor trainable with backpropagation. This study illustrates the usage in combining C2C-SN with a VAE (where the feature extractor is the encoder as shown in Figure 1) and the PIECE algorithm (where the feature extractor is a convolutional neural network) . If properly trained and regularized by the C2C-SN, the feature extractor would extract features that are both useful in its original downstream task (e.g. classification, reconstruction) and meaningful for inter-class analogies.

3.1. Training Process

The training process coordinates training of the C2C-SN and of the upstream ML feature extraction model, as shown in Algorithm 1. We highlight a few key points:

- Each batch of samples s_1 and the corresponding labels C_1 are used to calculate the original loss of the ML model as $L_f(f(s_1), C_1)$.
- A second batch of samples and labels is also created as s_2 and C_2 by shuffling s_1 and C_1 in the same order. A feature difference can be calculated as $f_\Delta = f(s_1) - f(s_2)$. A class label difference $C_\Delta = C_1 - C_2$ is constructed as the element-wise vector difference between the one hot encodings of C_1 and C_2 [32]. (f_Δ, C_Δ) serve as the positive training data (X_{pos}, y_{pos}) for the C2C-SN. Then C_2 is again shuffled into $\hat{C_2}$ so that every value is different and $\hat{C_2}$ is a batch of *wrong* labels for samples s_2. (f_Δ, $\hat{C_\Delta} = C_1 - \hat{C_2}$) will serve as the negative training data (X_{neg}, y_{neg}) for the C2C-SN.
- The positive and negative training data are combined into one set ($X = X_{pos} + X_{neg}, y = y_{pos} + y_{neg}$) and shuffled. The C2C-SN is trained using contrastive loss $L_{c2c}(X, y)$ to best predict whether the input sample difference f_Δ matches with the analogy of C_Δ.

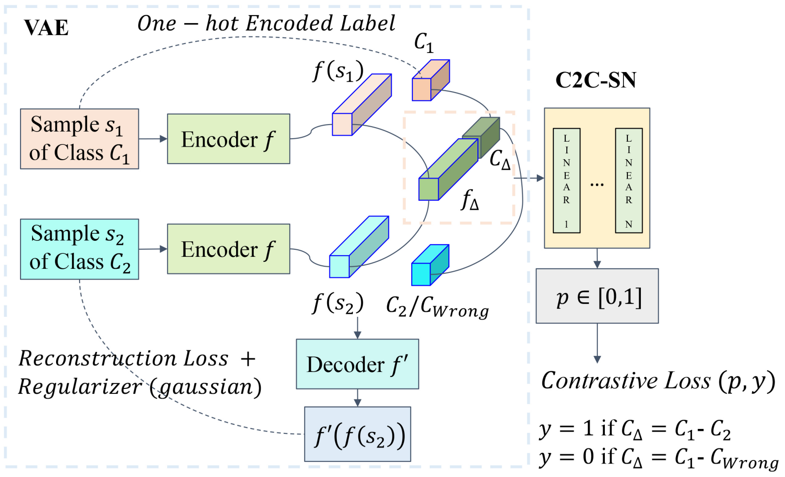

VAE
One − hot Encoded Label
C_1
$f(s_1)$
Sample s_1 of Class C_1
Encoder f
C_Δ
f_Δ
Sample s_2 of Class C_2
Encoder f
$f(s_2)$
C_2/C_{Wrong}
C2C-SN
LINEAR 1
...
LINEAR N
$p \in [0,1]$
Reconstruction Loss + Regularizer (gaussian)
Decoder f'
$f'(f(s_2))$
Contrastive Loss (p, y)
$y = 1$ if $C_\Delta = C_1$- C_2
$y = 0$ if $C_\Delta = C_1$- C_{Wrong}

Figure 1: Model Structure. Given a pair of samples s_1 and s_2 of class C_1 and C_2 respectively, the feature extractor f (a VAE's encoder in this case) extracts features $f(s_1)$ and $f(s_2)$. The feature difference $f_\Delta = f(s_1) - f(s_2)$ is concatenated with a label difference C_Δ (that is either correct C_1-C_2 or incorrect C_1-C_{wrong} where $C_{wrong} \neq C_2$). The expected output y of the C2C-SN is 1 or 0 depending on whether C_Δ is correct or not.

- Last, the training of the upstream model f and the C2C-SN can be combined by optimizing their parameters together to lower the total loss

$$L = L_f(f(s_1), C_1) + \alpha * L_{c2c}(X, y) \tag{1}$$

  where the parameter α controls the relative importance of the constrastive loss L_{c2c}.

3.2. C2C-SN Feature Importance

Once the C2C-SN is trained along with the feature extractor f, we apply a permutation feature importance method [30] (as explained in Molnar [31]) to calculate the importance of each value $f_{\Delta i}$ in the feature difference f_Δ. The method randomly permutes each value $f_{\Delta i}$ in f_Δ and measures how much the output of the C2C-SN is changed when using the new feature f'_Δ instead of the original feature f_Δ. If the value $f_{\Delta i}$ is important, then changing it will drastically change the final error of the C2C-SN. Because $f_{\Delta i}$ is calculated from $f_{\Delta i} = f(s_1)_i - f(s_2)_i$, the importance of $f_{\Delta i}$ is also the feature importance of $f(s_1)_i$ and $f(s_2)_i$ in the analogy C_1-C_2.

In contrast to other feature importance methods which permute one feature of the input and measure the variance in a model's output [30], the C2C-SN input includes both f_Δ and a class label difference C_Δ. Therefore C2C-SN feature importance can *rank features locally* based on C_Δ: The local feature important in one analogy can be of little significance in another analogy. The local feature importance is especially useful for counterfactual explanation by highlighting the most semantically meaningful difference between two classes. Samples of a class C_1 can be adapted into another class C_2 by changing the most important features in the analogy C_1-C_2, yielding a counterfactual. C2C-SN feature importance can also rank features as *global feature importance* across all C_Δ analogies.

Algorithm 1 Training Process of C2C-SN

```
Require: a feature extractor $f$ and its loss functions $L_f$ and a $c2c\text{-}sn$
  The following is repeated until model convergence or maximum epoch reached.
  for each batch of samples $s_1$ and labels $C_1 \in$ training data do
      $s_2, C_2 \Leftarrow shuffle\_together(s_1, C_1)$        ▷ Prepare a second batch from $s_1$ and $C_1$
      $f_\Delta \Leftarrow f(s_1) - f(s_2)$
      $C_\Delta \Leftarrow C_1 - C_2$
      $\hat{C_2} \Leftarrow shuffle(C_2)$        ▷ Prepare a batch of wrong labels $\hat{C_2}$
      $\hat{C_\Delta} \Leftarrow C_1 - \hat{C_2}$
      $X_{pos} \Leftarrow concat((f_\Delta, C_\Delta), dim = 1)$        ▷ Prepare the input and output for C2C-SN
      $y_{pos} \Leftarrow ones(size = X_{pos}.size)$
      $X_{neg} \Leftarrow concat((f_\Delta, \hat{C_\Delta}), dim = 1)$
      $y_{neg} \Leftarrow zeros(size = X_{neg}.size)$
      $X \Leftarrow concat((X_{pos}, X_{neg}), dim = 0)$
      $y \Leftarrow concat((y_{pos}, y_{neg}), dim = 0)$
      $X, y \Leftarrow shuffle\_together(X, y)$
      $L_{c2c}(X, y) \Leftarrow contrastive\_loss(c2c\text{-}sn(X), y)$
      $L = L_f(f(s_1), C_1) + \alpha * L_{c2c}(X, y)$
      Optimize $f$ and $c2c\text{-}sn$ over $L$
```

4. Experimental Study

We performed experiments to explore:

- Whether C2C-SN can extract features important for analogy and rank their importance.
- The quality of counterfactuals generated by modifying features selected by C2C-SN.
- The limitations of combining C2C-SN with a feature extractor.

4.1. Extracting and Ranking Features

Data set: The first experiment combines a C2C-SN with a VAE and tests it the MNIST and FashionMNIST data sets. MNIST is an image set of handwritten digits (from “0” to “9”) while FashionMNIST is an image set of 10 different types of clothing articles. All images are 28x28 pixels. Each data set contains 60000 training samples and 10000 testing samples.

Testbed systems: The VAE depends on meta-parameters. Most importantly, we set the channel number of convolutional layers $c = 64$ and the number of latent dimensions $latent\_dim = 32$. We chose these settings because they generally work well across multiple data sets. The VAE's encoder stacks two convolutional layers: ($in\_channels = 1$, $out\_channels = c$, $kernel\_size = 4$, $stride = 2$, $padding = 1$, $ReLU$) and ($in\_channels = c$, $out\_channels = c * 2$, $kernel\_size = 4$, $stride = 2$, $padding = 1$, $ReLU$). The features extracted by convolutional layers are passed to a linear layer to produce the embedding mu and another linear layer to produce std. The embedding mu is equivalent to the feature extracted $f(s)$ in Figure 1.

The decoder (marked as f' in Figure 1) is the reverse of the encoder: an embedding is passed to a linear layer ($in\_features = latent\_dims$, $out\_features = c * 2 * 7 * 7$), the output of which is passed to two convolutional layers: Conv2d ($in\_channels = c * 2$, $out\_channels = c$, $kernel\_size = 4$, $stride = 2$, $padding = 1$, $ReLU$) and Conv2d ($in\_channels = c$, $out\_channels = 1$, $kernel\_size = 4$, $stride = 2$, $padding = 1$, $Sigmoid$). As in standard VAEs, the encoder f and decoder f' together are trained to optimize a loss function L_f that is composite of a reconstruction loss (so the reconstructed image $f'(f(s))$ is similar to the original input s) and a KL-divergence loss (so the distribution of mus conforms to a prior Gaussian distribution whose $mu = 0$ and $std = 1$).

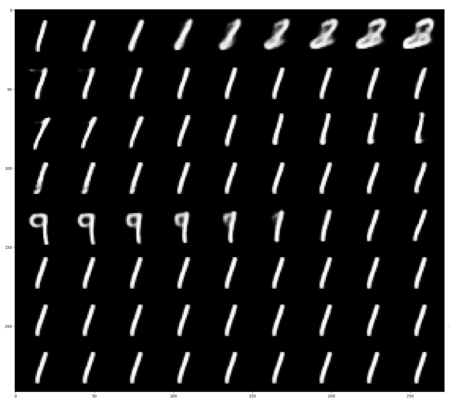
The C2C-SN is a composite of two hidden layers with leaky ReLU activation functions: ($in\_features = latent\_dims + label\_size$, $out\_features = c2c\_capacity$) and ($in\_features = c2c\_capacity$, $out\_features = c2c\_capacity/2$), and an output linear layer ($in\_features = c2c\_capacity/2$, $out\_features = 1$). The $c2c\_capacity$ is set to 32. The encoder f and the C2C-SN are trained together to minimize the constrastive loss L_{c2c}. Therefore, the encoder f, the decoder f' and the C2C-SN are all trained together to minimize the total loss $L = L_f + \alpha * L_{c2c}$ by following the procedure described in Section 3.1.

4.1.1. Experimental Results on MNIST

Given a sample s from MNIST, we use C2C-SN feature importance to rank features $f(s)$ both locally on analogical patterns between two classes and globally across all patterns. Using an image "1" as an example, Figure 2a shows the reconstructed images after modifying each of the top 8 globally ranked features out of 32 features in total. In Figures 2a-4b, each row shows the effect of modifying one feature for 9 equally spaced values. The values fall between -1 and 1, because the embeddings conform to a Gaussian distribution of $mu = 0$ and $std = 1$. The images in the center conform to the average of the class, while the ones on the two ends are more irregular, potentially of a different class. Figure 2a shows that features that are globally important may not necessarily lead to class change when modified. For example, modifying the 2nd - 4th most important features do not modify this "1" into digits of other classes.

C2C-SN feature importance does well at ranking features based on their local importance in the pattern between two specific classes. As shown in Figures 3a and 3b, C2C-SN feature importance correctly suggests the top features that can lead to the desired class change. More specifically, Figure 3a ranks features based on the analogy between "1"s and "2"s. It shows that the digit "1" can be modified into "2" by altering a single feature, however, too much modification might distort the image unrealistically, because the modified feature is now at the boundary of the distribution of latent embeddings. Figure 3b ranks features based on the pattern between "1"s and "9"s. It shows another example in which the digit "1" can be modified into a "9" by altering one single feature. Note that the digit "1" is modified into a "7" before it changes to a "9". This is because the embedding space of VAE is intrinsically smooth and other classes may be allowed between embeddings of two classes.

Figure 3c ranks features based on the pattern between "1"s and "5"s. Here, the C2C-SN failed to suggest any feature that can modify the "1" into a "5". Although the C2C-SN encourages the VAE encoder to extract features that are important in analogies, the training does not guarantee that a single feature corresponds to a given class change. Changing multiple features might be


(a) Gradual Changes of an Image "1"

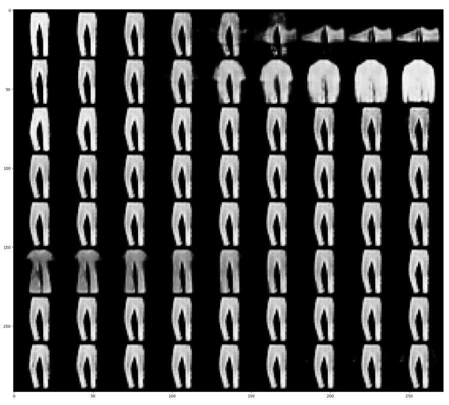
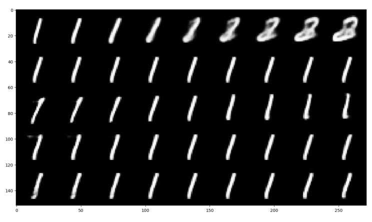
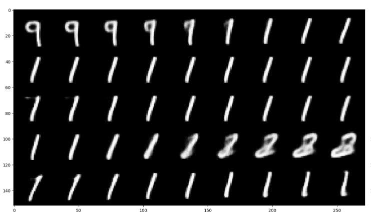

(b) Gradual Changes of a Pants Image

Figure 2: Gradual Changes of an Image when Modifying Top 8 Globally Ranked Features


(a)


(b)

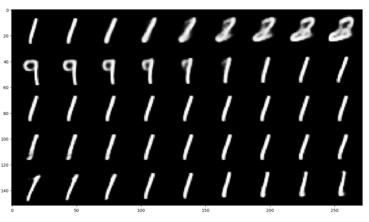

(c)

Figure 3: Gradual Changes of the Image "1" when Modifying Top 5 Locally Ranked Features in the Pattern from "1"s to (a) "2"s (b) "9"s (c) "5"s

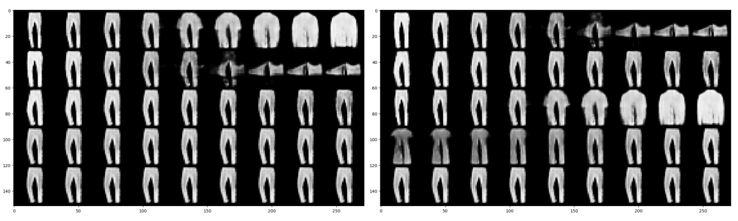
required for the change between two vastly different classes (e.g. "1"s and "5"s).

4.1.2. Experimental Results on FashionMNIST

The same experiment was carried out on FashionMNIST. Using a Pants image as an example, Figure 2b demonstrates the reconstructed images after modifying each of the top 8 globally ranked features. The majority of the features do not lead to class change when modified.

Figure 4a ranks features based on the pattern between Pants and Long Sleeves. Figure 4b ranks features based on the pattern between Pants and Shoes. Note that modifying one feature may not be enough to cause a desired change, as the Shoes converted from the Pants have a gap in the middle. This aligns with our earlier discussion of the results for MNIST.

These experiments illustrate that the C2C-SN can help a ML model to extract features for analogy and C2C-SN feature importance can rank these features for counterfactual explanation. However, it is not guaranteed that modifying a top feature will lead to desired class change, especially when the differences between the two classes are complex.



(a) (b)

Figure 4: Gradual Changes of a Pants Image when Modifying Top 5 Locally Ranked Features in the Pattern from Pants to (a) Long Sleeves (b) Shoes

4.2. Quality of Counterfactuals Using C2C-SN Feature Importance in PIECE

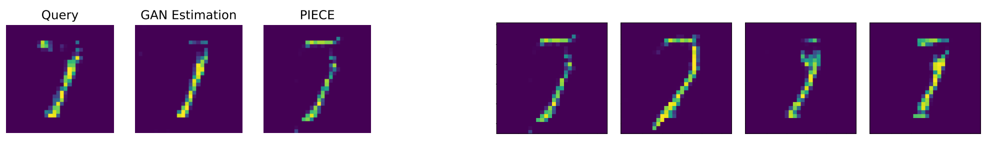
The strategy of learning analogies with C2C-SN should be generally applicable when using any feature extractor. This section presents an experiment combining C2C-SN feature importance with the PIECE algorithm [6], which uses a convolutional neural network for feature extraction.

This experiment generates counterfactuals for misclassified samples from Kenny and Keane [6] but replaces the feature selection component of PIECE with a new algorithm that selects top features ranked by C2C-SN feature importance. The experiment applies different methods to generate counterfactual explanations for MNIST and compares the generated images both qualitatively and quantitatively. The original design of PIECE has four major steps [6]:

1. A convolutional neural network cnn is trained to classify MNIST samples. We find 41 hard samples misclassified by cnn. We denote a hard sample image as s, the true class label as C_{true} and the misclassified label as C_{pred}.
2. The CNN derives the feature vector x of a sample s as the activation of the penultimate layer of cnn when applied to s, denoted as $x = cnn(s)$. A GAN-inversion technique is used to generate an image using a GAN G. A GAN embedding z for image s is optimized so that $x = cnn(G(z))$.
3. The probability function of each feature value x_i is modeled as a hurdle model. A feature value x_i is considered exceptional if the value is rare in the true class C_{true} (probabilities are less than 0.05). PIECE only considers modifying x_i if doing so will increase the output activation for C_{true}. Assume n feature values are to be modified by PIECE.
4. All n feature values are modified into their expected value in C_{true}. The modified feature is denoted as x'. Last, z is optimized into z' such that $cnn(G(z')) \approx x'$. The counterfactual explanation is $G(z')$.

Two sample counterfactuals are shown in Figures 5a and 6a. In both scenarios, the CNN classifier misclassifies a digit "7" is as a class other than "7", and the PIECE algorithm attempts to generate a counterfactual explanation by making the exceptional features normal. The modified image looks more like a standard "7". This explanation can be understood as follows: The model misclassified the query as non-"7" because it believes that the image needs to be like the counterfactual example to be considered a "7".

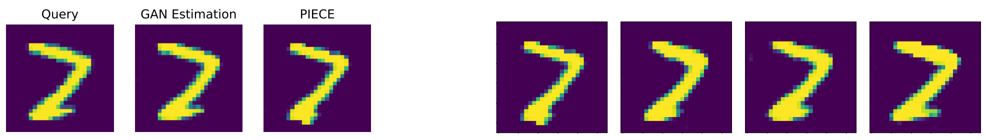
We first created a variation of the above algorithm (which we name as "C2C" for convenience), in which we replaced Step 3 in PIECE with the following: A C2C-SN is trained on the features



(a) A "7" Misclassified as "1". PIECE Modifies it into a "7".

(b) PIECE (c) C2C (d) P-STEP (e) C-STEP

Figure 5: Comparison of Four Different Counterfactuals Generated for the Digit "7" in Figure 5a



(a) A "7" Misclassified as "2". PIECE Modifies it into a "7".

(b) PIECE (c) C2C (d) P-STEP (e) C-STEP

Figure 6: Comparison of Four Different Counterfactuals Generated for the Digit "7" in Figure 6a

extracted and provides a C2C-SN feature importance ranking for the pattern C_{true}-C_{pred}. The top n features are then modified in their expected value in Step 4. The choice of n depends on how many features are deemed exceptional by PIECE. This variation selects features based on the C2C-SN feature importance ranking but modifies features in the same manner as PIECE.

We also made a variation of PIECE (named "PIECE-step") that, instead of modifying all exceptional features, modifies exceptional features one by one, starting from the most exceptional, until the modified sample is classified as C_{true} by the cnn. Last, we made a variation of C2C (named "C2C-step") which instead of modifying n features, modifies top features suggested by the C2C-SN feature importance one by one, until the modified sample is classified as C_{true}.

Thus we tested four variations: The original PIECE algorithm, PIECE-step that modifies features one by one until crossing the class boundary, the C2C algorithm that replaces PIECE's Step 3 with C2C-SN feature importance ranking, and the C2C-step that modifies features one by one until crossing the class boundary. The four were run on all 41 hard samples. Figures 5 and 6 show counterfactual examples generated by different algorithms for two sample digit "7"s. We observe that PIECE and C2C both produce plausible counterfactuals, and their corresponding step variations also produce counterfactuals near the class boundary.

We compared the models quantitatively, reusing the following metrics the original evaluation of PIECE [6]: IM1 and IM2 are two measures of the interpretability of counterfactuals [33]; The posterior mean (MC-Mean) and posterior standard deviation (MC-STD) of MC dropout measures the model's confidence and uncertainty on the counterfactuals [34]; Last, NN-Dist measures the distance between a counterfactual and a nearest neighboring training sample. These metrics quantitatively reflect how realistic the generated images are. Table 1 compares the counterfactuals generated by different algorithms for the 41 hard samples under the five metrics. Min-Edit is included as a baseline counterfactual generation algorithm. Note that PIECE and C2C are comparable to each other because they modify a fixed number n of features, while the rest of the algorithms modify features one by one until crossing the class boundary.

	MC-Mean	MC-STD	NN-Dist	IM1	IM2
PIECE	**90591**	248393	**0.411**	3.005	0.064
C2C	5937	**12574**	0.626	**2.338**	0.064
Min-Edit	**600**	1168	1.066	3.897	**0.067**
PIECE-step	395	**668**	1.029	**2.227**	0.072
C2C-step	409	811	**1.014**	2.607	0.073
which is better?	higher	lower	lower	lower	lower

Table 1
Comparison of Counterfactual Generation Algorithms

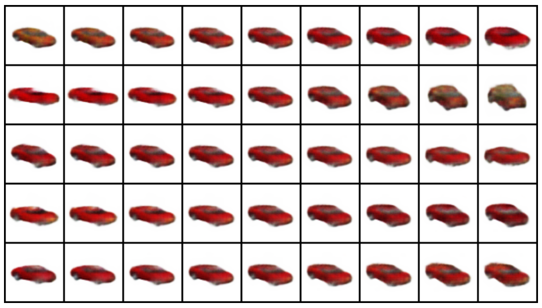


Figure 7: Gradual Changes of a Car's Image when Modifying Top 5 Locally Ranked Features in the Inter-Elevation Pattern

Both the qualitative and quantitative comparisons show that PIECE and C2C (PIECE-step and C2C-step correspondingly) generate counterfactuals with their own unique "tastes" but achieve similar overall performance.

4.3. Limitations of Analogies Found by C2C-SN

The assumption that there exists an analogy between two classes C_1 and C_2 may not hold when either class is widely dispersed. For x_1, x_2 of class C_1 and y_1, y_2 of class C_2, x_1 (or y_1) can be very different from x_2 (or y_2) and the feature difference $f(x_1) - f(y_1)$ can be not consistent with $f(x_2) - f(y_2)$. We demonstrate this issue in the following experiment combining C2C-SN with a Guided-VAE [35] on the Cars3d data set [11].

Cars3d is a multi-label data set where each sample is an image associated with one of 184 car models, one of 24 azimuths and one of 4 camera elevations. Feature entanglement is intrinsically difficult in the Cars3d data set. For example, two images of different car models may pose with different camera angles, obscuring the analogy between the models. We experimented with Guided-VAE which reportedly encourages feature disentanglement through supervised learning [35]. However, features are still entangled. Figure 7 shows changes in features extracted for inter-elevation pattern often change the azimuth or car model as a side-effect.

The sub-optimal experimental result can be explained in a few ways: (1) C2C-SN encourages features extracted to be relevant for a target analogy but does not prevent the entanglement of other analogies. (2) C2C-SN relies on the feature extractor. If the feature extractor fail to disentangle features, C2C-SN suffers with it. (3) The analogical pattern between two complex classes may involve multiple features instead of one.

As a future direction, an analogical suitability test can be devised to check whether a data set with a feature extractor is suitable for analogy learning using C2C-SN.

5. Conclusion

This paper proposes a generally applicable method to extract analogies corresponding to general similarity and difference patterns between elements of classification domains, and illustrates the usefulness of such analogies in counterfactual explanation. An ML model with a feature extractor and a C2C-SN can be trained together where the loss of the C2C-SN functions as a regularizer for the original ML loss. The ML model is thus trained to extract features that are useful for (1) the original purpose of the ML model (e.g. classification or image reconstruction) and (2) describing analogies. This method can provide C2C-SN feature importance rankings and suggest features to modify for counterfactual explanations. Experimental results of quantitative evaluations show results comparable to an existing method while the proposed method is more generally applicable. An interesting future question is to apply C2C-SN to discover analogies in other domains such as natural language texts.

Acknowledgments

This paper is adapted from "Selecting Feature Changes for Counterfactual Explanation: A Class-to-Class Approach," which appeared in the non-archival proceedings of the IJCAI 2023 Workshop on Explainable AI (XAI).

References

[1] R. P. Hall, Computational approaches to analogical reasoning: A comparative analysis, Artificial Intelligence 39 (1989) 39–120. URL: https://www.sciencedirect.com/science/article/pii/0004370289900039. doi:https://doi.org/10.1016/0004-3702(89)90003-9.

[2] D. Hofstadter, M. Mitchell, The copycat project: A model of mental fluidity and analogy-making, in: K. Holyoak, J. Barnden (Eds.), Advances in Connectionist and Neural Computation Theory, volume 2, Ablex, 1993, pp. 31–11.

[3] S. E. Reed, Y. Zhang, Y. Zhang, H. Lee, Deep visual analogy-making, in: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 28, Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/e07413354875be01a996dc560274708e-Paper.pdf.

[4] N. Ichien, Q. Liu, S. Fu, K. J. Holyoak, A. L. Yuille, H. Lu, Visual analogy: Deep learning versus compositional models, Proceedings of the 43rd Annual Meeting of the Cognitive Science Society (2021). URL: https://par.nsf.gov/biblio/10231806.

[5] E. Hüllermeier, Towards analogy-based explanations in machine learning, in: V. Torra, Y. Narukawa, J. Nin, N. Agell (Eds.), Modeling Decisions for Artificial Intelligence, Springer International Publishing, Cham, 2020, pp. 205–217.

[6] E. M. Kenny, M. T. Keane, On generating plausible counterfactual and semi-factual explanations for deep learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 11575–11585.

[7] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, S. Lee, Counterfactual visual explanations, in: International Conference on Machine Learning, PMLR, 2019, pp. 2376–2384.

[8] D. P. Kingma, M. Welling, Auto-Encoding Variational Bayes, in: 2nd International Conference on Learning Representations, ICLR 2014, 2014.

[9] Y. LeCun, C. Cortes, MNIST handwritten digit database (2010). URL: http://yann.lecun.com/exdb/mnist/.

[10] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL: http://arxiv.org/abs/1708.07747, arxiv:1708.07747.

[11] S. E. Reed, Y. Zhang, Y. Zhang, H. Lee, Deep visual analogy-making, in: Advances in Neural Information Processing Systems, volume 28, Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/e07413354875be01a996dc560274708e-Paper.pdf.

[12] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a "siamese" time delay neural network, in: Proceedings of the 6th International Conference on Neural Information Processing Systems, NIPS'93, Morgan Kaufmann, San Francisco, 1993, pp. 737–744.

[13] X. Ye, D. Leake, W. Huibregtse, M. Dalkilic, Applying class-to-class siamese networks to explain classifications with supportive and contrastive cases, in: Case-based reasoning research and development, ICCBR-20, Springer, 2020, pp. 245–260.

[14] M. Bounhas, H. Prade, G. Richard, Some recent advances in reasoning based on analogical proportions, 2022. `arXiv:2212.11717`.

[15] M. Bounhas, H. Prade, G. Richard, Analogy-based classifiers for nominal or numerical data, International Journal of Approximate Reasoning 91 (2017) 36–55. URL: https://www.sciencedirect.com/science/article/pii/S0888613X17305303. doi:`https://doi.org/10.1016/j.ijar.2017.08.010`.

[16] M. Couceiro, N. Hug, H. Prade, G. Richard, Behavior of analogical inference w.r.t. boolean functions, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, IJCAI, 2018, pp. 2057–2063.

[17] E. Marquer, F. Badra, M.-J. Lesot, M. Couceiro, D. Leake, Less is better: An energy-based approach to case base competence, in: ATA'23: Workshop on Analogies: From Theory to Applications, ICCBR 2023 Workshop Proceedings, CEUR, 2023, pp. 27–42.

[18] S. Lim, H. Prade, G. Richard, Using analogical proportions for explanations, in: F. Dupin de Saint-Cyr, M. Öztürk-Escoffier, N. Potyka (Eds.), Scalable Uncertainty Management, Springer International Publishing, Cham, 2022, pp. 309–325.

[19] H. Prade, G. Richard, Analogical proportions: Why they are useful in AI, in: International Joint Conference on Artificial Intelligence, IJCAI, 2021, pp. 4568–4576.

[20] M. Mitchell, Analogy-Making as Perception, MIT Press/Bradford Books, Cambridge, MA, 1993.

[21] S. Hu, Y. Ma, X. Liu, Y. Wei, S. Bai, Stratified rule-aware network for abstract visual reasoning, in: AAAI Conference on Artificial Intelligence, AAAI, 2020, pp. 1567–1574.

[22] E. Marquer, M. Couceiro, Solving morphological analogies: from retrieval to generation,

ArXiv abs/2303.18062 (2023). URL: https://api.semanticscholar.org/CorpusID:257900745.

[23] C. Allen, T. M. Hospedales, Analogies explained: Towards understanding word embeddings, in: International Conference on Machine Learning, PMLR, 2019, pp. 223–231.

[24] M. Mitchell, Abstraction and analogy-making in artificial intelligence, Annals of the New York Academy of Sciences 1505 (2021). URL: https://api.semanticscholar.org/CorpusID:231985717.

[25] S. Joshi, O. Koyejo, W. D. Vijitbenjaronk, B. Kim, J. Ghosh, Towards realistic individual recourse and actionable explanations in black-box decision making systems, ArXiv abs/1907.09615 (2019). URL: https://api.semanticscholar.org/CorpusID:198179624.

[26] C. Theobald, F. Pennerath, B. Conan-Guez, M. Couceiro, A. Napoli, Clarity: an improved gradient method for producing quality visual counterfactual explanations, 2022. arXiv:2211.15370.

[27] M. T. Ribeiro, S. Singh, C. 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, Association for Computing Machinery, New York, NY, USA, 2016, p. 1135–1144. URL: https://doi.org/10.1145/2939672.2939778. doi:10.1145/2939672.2939778.

[28] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 4768–4777.

[29] S. Mishra, S. Dutta, J. Long, D. Magazzeni, A survey on the robustness of feature importance and counterfactual explanations, 2023. arXiv:2111.00358.

[30] A. Fisher, C. Rudin, F. Dominici, All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously, 2019. arXiv:1801.01489.

[31] C. Molnar, Interpretable Machine Learning, 2 ed., 2022. https://christophm.github.io/interpretable-ml-book.

[32] X. Ye, D. Leake, D. Crandall, Case adaptation with neural networks: Capabilities and limitations, in: Case-Based Reasoning Research and Development ICCBR-22, Springer, Cham, 2022, pp. 143–158.

[33] A. Van Looveren, J. Klaise, Interpretable counterfactual explanations guided by prototypes, in: Machine Learning and Knowledge Discovery in Databases. Research Track, Springer, Cham, 2021, pp. 650–665.

[34] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, 2016, p. 1050–1059.

[35] Z. Ding, Y. Xu, W. Xu, G. Parmar, Y. Yang, M. Welling, Z. Tu, Guided variational autoencoder for disentanglement learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7920–7929.