ECINN: Efficient Counterfactuals from Invertible Neural Networks

Frederik Hvilshøj[1,3]
fhvilshoj@cs.au.dk
Alexandros Iosifidis[2,3]
ai@ece.au.dk
Ira Assent[1,3]
ira@cs.au.dk

[1] Dept. of Computer Science
Aarhus University, Denmark
[2] Dept. of Electrical and Computer Engineering
Aarhus University, Denmark
[3] DIGIT centre, Aarhus University

Abstract

Counterfactual examples identify how inputs can be altered to change the predicted class of a classifier, thus opening up the black-box nature of, *e.g.*, deep neural networks. We propose a method, ECINN, that utilizes the generative capacities of invertible neural networks for image classification to generate counterfactual examples efficiently. In contrast to competing methods that sometimes need a thousand evaluations or more of the classifier, ECINN has a closed-form expression and generates a counterfactual in the time of only two evaluations. Arguably, the main challenge of generating counterfactual examples is to alter only input features that affect the predicted outcome, *i.e.*, class-dependent features. Our experiments demonstrate how ECINN alters class-dependent image regions to change the perceptual and predicted class, producing more realistically looking counterfactuals three orders of magnitude faster than competing methods.

1 Introduction

A great effort has been devoted to open up the black-box nature of deep neural networks for computer vision. Among others, heatmaps [3], class-maximizing samples [29], and contrastive examples [9] have been proposed; we focus on the latter. Contrastive examples are also known as counterfactual examples, even though models do not possess any causal structure as described in [26].[1] We adopt the setting from [13] and consider the generic question, "For situation X, why was the outcome Y and not Z?" We provide a counterfactual example to give an explanation of the form "Had X been $\hat{X}$, then the outcome would have been Z."

Being able to provide counterfactual examples for complex neural networks has an immense potential to improve human-model-interactions. To name but a few, surveillance systems could be assessed for biases when picking out candidates for screening and self-driving vehicles could be better diagnosed when misinterpreting their image feeds [13].

[1] Counterfactual examples as described in [26] are based on structured causal graphs relating inputs and outputs. In the image domain, it is generally not known how to make such graphs and a causal analysis is thus not possible.

We propose Efficient Counterfactuals from Invertible Neural Networks (ECINN), which utilizes Invertible Neural Network (INN) classifiers to generate counterfactual examples. Figure 1 depicts the high-level structure of ECINN. An image of a woman *without* makeup (left) is transformed by an INN denoted f into an internal representation (center). The internal representation is corrected, as indicated by the green arrow, before being reverted by f^{-1} to form a counterfactual example *with* makeup (right).

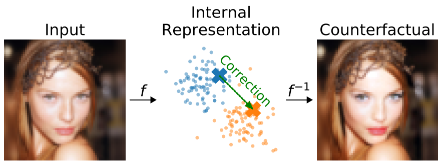


Figure 1: f transforms image *without* makeup (left) into internal representation which is corrected with closed-form expression (center). f^{-1} generates counterfactual example *with* makeup (right).

The properties of INNs make a one-pass-solution possible. In contrast to usual discriminative models, INNs are known to have semantically organized latent spaces where translations in specific directions result in semantic changes in the input space [11]. Importantly, INNs even have full information-preservation between input and output layers in contrast to, *e.g.*, auto-encoders [6], which allows exact recovery of inputs from outputs. As such, it can be argued that INNs are ideal for combining generative and discriminative capabilities [4].

Existing methods for generating counterfactual examples [1, 8, 9, 12, 13, 15, 25, 31, 32, 35] need to query the model under consideration many times due to various numerical optimization algorithms, obtain non-unique counterfactual examples, and need hyperparameter tuning. To the best of our knowledge, we introduce the first algorithm which uses only one forward and reverse pass, produces unique counterfactual examples, and needs no hyperparameter tuning. ECINN is even fast enough to be used in an interactive setting [7].

Good counterfactual examples are broadly agreed to be realistic, minimal, and actionable [12, 33]. In the image domain, however, minimal changes are hard to quantify in a semantically meaningful way. As such, we argue that the main challenge is to generate *realistically* looking images with *perceptible* changes only to class-relevant features.

We demonstrate experimentally how ECINN produces counterfactual examples leaving class-independent features largely untouched while class-dependent features are changed successfully. Experiments also demonstrate that our theoretically derived one-pass-solution yields running times more than three orders of magnitudes faster than competing methods.

2 Related Work

Counterfactual Examples. Many methods have been proposed for generating counterfactual examples or identifying counterfactual features. To name but a few, [1, 13, 31, 32, 34] operate on image data, [15, 16, 35] consider text, and yet other methods operate on relatively low dimensional data compared to images and text [8, 12, 33].

Methods for generating counterfactual examples can be categorized by the insights needed into the predictive model. Methods from the first category consider the predictive model as opaque and need no insight. Methods from the second category utilize gradients of the predictive model, while methods from the last category use internal data representations of the predictive model. All methods mentioned need to query the predictive model many times. In contrast, after a preprocessing step that needs to be done only once, our method uses a single forward and inverse pass through the model to generate a counterfactual example.

In the first category, methods operating on opaque models iteratively generate candidate sets before querying the predictive model to test candidates. [12] utilizes a greedy heuristic from simple data statistics to determine what input features to perturb, while [28] uses a genetic algorithm. [32] segments input images into super-pixels and use a greedy algorithm to perturb super-pixels. On text data, [35] finetunes a GPT-2 model [27] to generate similar sentences to the input sentence to generate new candidates. [34] identifies counterfactual regions in input images but does not generate counterfactual examples.

The second category employs gradient optimization techniques to generate counterfactual examples. Albeit from a different perspective, previous work has developed methods for synthesizing inputs that maximize desired (output) neurons. For example, [29] uses gradient descent with an L_2-norm prior loss on a random input. [23] includes a local pixel variation prior to obtain more realistically looking features. [33] also proposed a different loss based on the median absoute deviation. Even though the methods give insights into the inner workings of the classifier, they suffer from generating unrealistic images. More recently, [25] proposed to train a generative model to generate counterfactual examples. In a similar vein, [9] utilizes a pretrained and fixed auto-encoder to identify latent codes that generate desired outputs through gradient optimization. An extension of [9] is [31] which uses auto-encoders or KD-trees to identify class prototypes which helps guide the gradient optimization. In comparison to our one-pass-solution, the default maximum queries of the classifier in the official code of [31] is 1000.[2] Finally, [22] uses gradients of the classifier to train an external variational auto-encoder to generate counterfactuals fast. In contrast to ECINN, the method has a substantial pre-computation time due to training of the auto-encoder.

The third category of methods contains two different strategies. First, [13] considers convolutional neural networks as a composition of a (convolutional) feature extractor and a classification network. They propose two algorithms for mixing feature fibers of the input sample and a sample from the target class. Second, [1] similarly uses a part of the classifying network as a feature extractor to cluster features, yielding an identification of semantic features like stripes, wool, etc. A gradient descent algorithm successively adds or removes features from the input to obtain a counterfactual example.

Although conceptually different, our work fits best into the third category. Instead of generating counterfactual examples from an "arbitrary" neural network, we choose a specific family of neural networks, INNs, to generate counterfactual examples efficiently without the need for multiple queries of the model or memory consuming gradient computations.

INNs as generative classifiers. INNs have gained wide attention as unsupervised models which allow generating realistically looking "fake" samples [10, 11, 19], typically referred to as Normalizing Flows. Despite hidden in appendices, both [11] and [19] present samples generated from class-conditional INNs. Later, it was explicitly described how to compose INNs with a Gaussian mixture model (GMM) to obtain a generative classifier [14, 24]. However, adding classification abilities comes at a price. As demonstrated in [2], there is a trade-off between classification performance and the quality of the generated fake images. The work introduces an information bottleneck loss, which explicitly trades off the classification and generation performance through a hyperparameter β. [2] further introduces a new invertible model architecture, which we refer to as IB-INN.

Regarding interpretability, [21] shows how conditional INNs can be trustworthy classifiers by, *e.g.*, visualizing decision spaces and computing posterior heatmaps. Here, we further show conditional INNs to be trustworthy by using them for generating counterfactual.

[2]Official code: https://docs.seldon.io/projects/alibi/en/stable/api/alibi.explainers.cfproto.html

3 Efficient Counterfactual Examples

This section constitutes our main contribution. We combine theoretical insights and practical observations from INNs to generate unique counterfactual examples from just one forward and inverse pass without the use of any numerical optimization techniques.

3.1 Problem Statement

As mentioned, counterfactual examples indicate why an input was predicted to be one class rather than another. Specifically, we adopt the definition from [33] which states that counterfactual examples are statements taking the form: “Score p was returned because variables V had values $(v_1, v_2, \dots)$ associated with them. If V instead had values $(v'_1, v'_2, \dots)$, and all other variables had remained constant, score p' would have been returned.” In the context of image classification, counterfactual examples are visualizations showing how the input image can be altered to change the predicted class.

Desiderata. In line with the desiderata of [12] and [33], we find that three properties are critical for counterfactuals to be useful. i) *Only semantically relevant features should be changed*. For example, facial features like lips and cheeks might change while the background should not when a counterfactual is generated for a face without makeup. ii) *Counterfactuals should look realistic*. Unrealistic counterfactuals might have misplaced eyes, extreme color values, or a “one-pixel-change” like the adversarial examples presented in [30]. iii) *Tipping-point counterfactuals and convincing counterfactuals should be prioritized*. With tipping-point, we refer to counterfactuals on the decision boundary, just where the prediction changes from the input to the target class and convincing counterfactuals are samples beyond the decision boundary that gets high probabilities for the target class.

Definition 1. *(tipping-point counterfactual) Given a classifier with posterior probabilities $p(y|x)$, an input $x \in \mathcal{X}$, and a predicted class $y = \arg\max_y p(y|x)$, a counterfactual $\hat{x}^{(q)}$ with target class q is a tipping-point counterfactual if there exists a path $h : [0;1] \to \mathcal{X}$ and constant $C \in]0;1[$ such that $h(0) = x$; $h(C) = \hat{x}^{(q)}$; for $c < C$, y has higher probability than q,* i.e., *$p(y|h(c)) > p(q|h(c))$; for $c = C$, probabilities are equal,* i.e., *$p(y|h(c)) = p(q|h(c))$; and for $c > C$, q has higher probability than y,* i.e., *$p(y|h(c)) < p(q|h(c))$.*

Definition 2. *(convincing counterfactual) given classifier $p(y|x)$ for K classes and input x as defined above, a counterfactual $\hat{x}^{(q)}$ is a convincing counterfactual if*

$$\forall y' : y' \in \{1, \dots, K\} \setminus \{q\} \wedge p(q|\hat{x}^{(q)}) \gg p(y'|\hat{x}^{(q)}).$$

Tipping-point counterfactuals are essential because they represent minimal corrections to the input. However, they might not always make sense due to visual class differences. For example, when changing the predicted class of a cat to a dog, a tipping-point counterfactual might mix pointy and hanging hears because it is situated at the decision boundary. On the contrary, a convincing counterfactual would successfully show such transformation, but potentially with overly pronounced changes. Providing both types of explanations thus give a deeper insight into the decisions of the classifier.

In the supplementary material, we prove that ECINN produces valid tipping-point counterfactuals according to Definition 1 and in the experiments (Section 4), we verify that ECINN also complies with Definition 2 and the remaining desiderata.

3.2 Conditional INNs

We find INNs to be well suited for the counterfactual problem because they are bijective, *i.e.*, every latent vector corresponds to exactly one input. In contrast, typical classification models are inherently surjective, *i.e.*, there potentially exist many inputs which produce each output. In turn, INNs admits a single inverse pass to perfectly identify the right input while surjective models must rely on approximate solutions from less efficient numerical methods.

It is known that well-trained INNs have semantically organized latent spaces [1]. We believe that when many latent representations of samples from the same class are averaged, then class-independent information like background and object orientation will cancel out and leave just class-dependent information. ECINN isolates such latent class-dependent information to correct embeddings for generating counterfactual examples.

A conditional INN f is typically trained by computing latent vectors $z = f(x)$ from input vectors x and using the latent vectors to fit a GMM to class labels y. However, to use z rather than x in the GMM, one must use the change-of-variables formula, which states that

$$\log p_X(x|y) = \log p_Z(f(x)|y) + \log |det(J)|. \tag{1}$$

That is, the class-conditional log density of an input x in the image space, $p_X(x|y)$, is equal to the class-conditional log density of $f(x)$ in the latent space $p_Z(f(x)|y)$, but with an additional Jacobian term, $J = \frac{\partial f(x)}{\partial x}$. We choose class-dependent latent densities to be Gaussians, $p_Z(z|y) = \mathcal{N}(\mu_y, \mathbb{1})$. By Bayes' rule, we notice that under a uniform prior distribution over labels, $p(y) = \frac{1}{K}$ for K classes, the log posterior probability becomes

$$\log p_X(y|x) = \log \frac{p_X(x|y)}{\sum_{y'} p_X(x|y')} \propto -||f(x) - \mu_y||^2. \tag{2}$$

From Equation (2), we see that independent of the Jacobian determinant, latent vector $z = f(x)$ will be predicted to be from the class y with the closest model mean, μ_y. In turn, the latent space of the classifier can be analyzed under L_2-norms instead of less efficient and complex densities $p_X(x|y)$, which depend on the Jacobian determinant. In the following, we present how ECINN utilizes this insight to produce counterfactual examples efficiently.

3.3 ECINN

At a high level, ECINN transforms images into a latent space through an INN f. In the latent space, a closed-form expression changes the predicted class by correcting the embedding. From the corrected embedding, a counterfactual is generated by the inverse INN f^{-1}.

As a preprocessing step that needs to be done only once and takes just five seconds on MNIST, we group the training samples by their classified output, $G_j = \{x|C(x) = j\}$, where $C(x) = \arg\max_y p_X(y|x)$ is the predicted class. Afterwards, we compute mean latent vectors $\bar{\mu}_j = \frac{1}{|G_j|} \sum_{x \in G_j} f(x)$ for each class j and define the vector from $\bar{\mu}_p$ to $\bar{\mu}_q$ as $\Delta_{p,q} = \bar{\mu}_q - \bar{\mu}_p$.

Given a target class q and an input x, a counterfactual example $\hat{x}^{(q)}$ is produced from the predicted class $C(x) = p$ by adding a scaled version of $\Delta_{p,q}$ to the latent space embedding $z = f(x)$ and inverting it through the INN,

$$\hat{x}^{(q)} = f^{-1}(f(x) + \alpha \Delta_{p,q}). \tag{3}$$

It follows that a counterfactual example requires just one evaluation of f and f^{-1}.

To follow our third desideratum and provide both tipping-point and convincing counterfactuals, we compute two counterfactuals for each input with different values of α. First, we choose α_0 to produce a tipping-point counterfactual. Due to Equation (2), α_0 is identified analytically such that $||z+\alpha_0\Delta_{p,q}-\mu_p|| = ||z+\alpha_0\Delta_{p,q}-\mu_q||$, which moves the latent vector to the decision boundary between the input and target class. The closed-form expression for α_0 is derived in the appendix (Section A), along with a proof that it complies with Definition 1 (Section B). Second, we choose α_1 such that the target class q is predicted with high confidence to produce a convincing counterfactual. α_1 is chosen heuristically to be $\alpha_1 = \alpha_0 + \frac{4}{5}(1-\alpha_0)$ (see supplementary material for details). Although not guaranteed that $C(\hat{x}^{(q)}) = q$, we observe that the relation holds in practice.

In Figure 2, we illustrate the intuition of ECINN. The figure shows two isotropic Gaussians in the latent space with a blue decision boundary. With green empty squares, we indicate the two computed means $\bar{\mu}_p$ and $\bar{\mu}_q$, connected by $\Delta_{p,q}$ (green arrow). The orange line passes through z in direction $\Delta_{p,q}$. The two points of interest are the blue square on the intersection of the blue and the orange line and the black square to the right. According to the model, the blue square constitutes a tipping-point counterfactual, and the black square is very likely to stem from class q, *i.e.*, a convincing counterfactual.

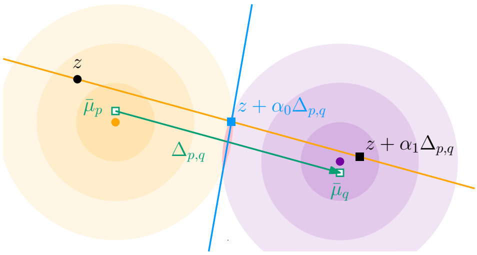


Figure 2: Latent space corrections by ECINN.

In conclusion, we introduce ECINN which allows computing counterfactuals efficiently by utilizing theoretical and observational properties of INNs. ECINN complies with our first two desiderata by generating counterfactuals which represent class-dependents changes while leaving out most class-independent information. By providing both tipping-point and convincing counterfactuals, it also follow the third desideratum.

4 Experiments

In this section, we evaluate how our counterfactual examples perform. Our experiments show how ECINN produces meaningful counterfactual examples across three different image datasets, changes class-dependent features while maintaining class-independent features, and outperforms competing methods.

Experimental Details. We evaluate ECINN on a synthetic FakeMNIST dataset, the MNIST dataset [20], and the CelebA-HQ dataset [17]. On all three datasets, classification errors of the IB-INN models are comparable to those of a standard classification network (see Table 2 in the appendix). For all our experiments, we have trained IB-INN models "as-is."[3] We found that β-values for IB-INN close to one strike a good balance between classification accuracy and generative performance (see appendix). We also provide an overview of hardware, all models used, hyperparameters, and the model performances in the appendix along with additional samples of all plots in the supplementary material. Results presented are all with samples from the test sets and were found to be consistent across samples.

[3]We adopted models and training code from https://github.com/VLL-HD/IB-INN.

(a) Random samples. (b) Counterfactual examples.

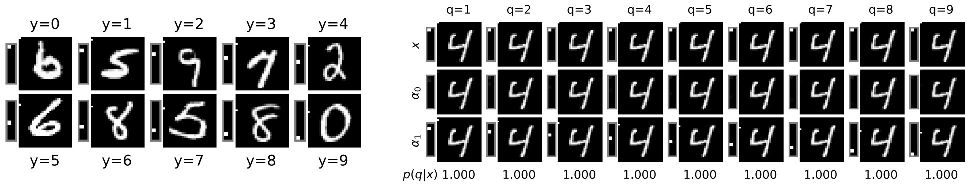


Figure 3: FakeMNIST dataset. For improved readability, smaller rectangles to the left of images magnify the top left 10×2 pixels, indicating the class.

We provide code for training IB+INN models and and explaining them with ECINN at https://github.com/fhvilshoj/ECINN.

4.1 FakeMNIST

To verify ECINN in a controlled setting, we carefully design a dataset such that less than two percent of the pixels in each image are class-*dependent*. As argued, a proper counterfactual example for a well-trained model should alter only the class-dependent pixels and if no class-dependent information is present, each class should be equally likely.

The dataset is generated by randomly reassigning labels to images. We alter images *only* by injecting the information of the new labels in the top-left 10×1 pixels; the ith top-left pixel will be white if the images is labeled i. For example, if an image gets label "5," the sixth pixel in the left column is white. Figure 3a shows a sample from each of the ten classes. Only the top-left pixels depends on the labels; the depicted digits do not.

Figure 3b shows random sample from the class $y = 0$ (first row) and tipping-point counterfactuals (α_0) in the second row. The third row includes convincing counterfactual examples (α_1). Each column corresponds to a different target class q. Figure 3b shows that the dot in the top left corner of the input does change position, while the class-independent digit remains unchanged as expected. Specifically, the third row from left to right reveals how the dot in the top left corner travels downwards to end in the tenth pixel. The second row has no dot, which aligns well with the interpretation about equally likely class probabilities above.

4.2 MNIST

Next, we apply ECINN to the MNIST dataset. First, we verify our second desideratum, *i.e.*, that ECINN produces realistic counterfactual examples. Second, we investigate how well class-independent features like font-weight and tilt are maintained by ECINN, *i.e.*, our first desideratum. Finally, we compare ECINN to two competing methods.

Realistic Counterfactuals. In Figure 4, we depict counterfactual examples in the same fasion as Figure 3b. The figure shows how an image of a three is properly transformed into any of the remaining nine classes. Note that in the second row, the counterfactual examples are in many cases

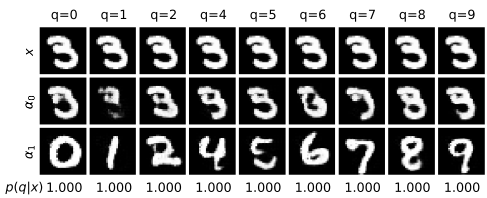
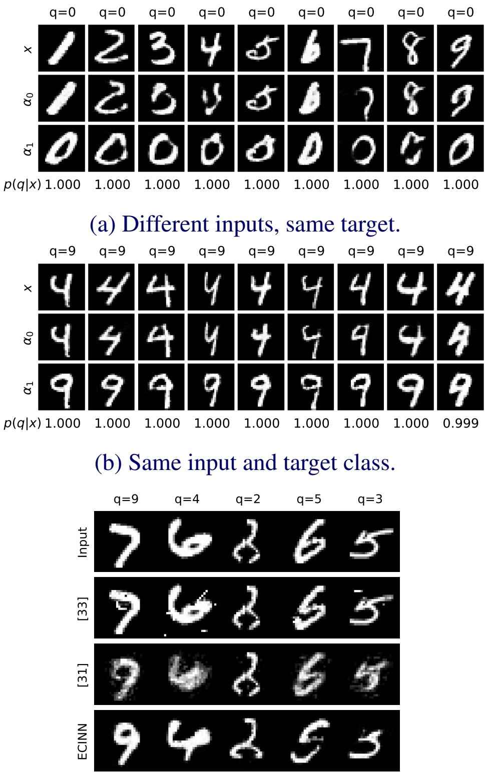


Figure 4: Same input, different targets.

such that even a human might mistake the image for both the input and target class. By contrast, the third row contains samples where the three has successfully transformed into the target class. This experiment demonstrates that ECINN complies with Definition 2 ($p(q|x) = 1$ for all samples) and our second desideratum by generating realistic counterfactuals.

Class-Independent Properties. In Figure 5a and 5b, we demonstrate how class-independent properties like font-weight, tilt, and size are preserved during counterfactual generation. First, Figure 5a includes nine different inputs (first row), each from a different class, that are all translated to the target class, $q = 0$. We observe that the nine outcomes (row three) are perceptually different while resembling the target class. Each counterfactual example maintains class-independent properties from the input while resembling the target class. For example, the narrow and tilted one (first column) becomes a narrow and tilted zero. The observation suggests that ECINN maintains properties that are not directly dependent on the label.

In Figure 5b, we investigate how class-independent properties are maintained. We sample nine different images from the class $y = 4$ and compute their counterfactual examples for the target class $q = 9$. We observe how bold inputs yield bold counterfactuals; likewise, slim inputs yield slim counterfactuals. Similar observations can be made for, *e.g.*, tilt, size, and shape.



(a) Different inputs, same target.

(b) Same input and target class.

(c) Comparison to [33] and [31].

Figure 5: Counterfactuals for MNIST.

Comparison. In Figure 5c, we compare counterfactual examples generated with the algorithms proposed in [33] and [31] with our method.[4] Rows correspond to counterfactual methods and columns represents five different inputs. The figure shows that both competing methods generate more artificially looking counterfactual examples than ECINN. As the figure also shows, we found across many samples that counterfactuals generated by [33] and [31] generally look more artificial by having disconnected white pixels and being blurred, respectively. See supplementary material for additional samples.

Mehod	Mean (std)	n
[33]	21.64 (7.99)	100
[31]	16.85 (0.35)	100
ECINN	**0.0025 (0.0002)**	10^4

Table 1: MNIST Computation times.

In Table 1, we compare computation time on a single GPU, similar to [6]. For a fair comparison, we do not batch samples, as the framework for the competing methods does not support batching. The table shows how ECINN is more than 6000× faster than competing

[4] Implementations found at https://github.com/SeldonIO/alibi; applied with default parameters.

methods. As it takes significaltly less time than 0.1 second, ECINN can even be used in an interactive setting [7], which is not possible with these competing methods.

In conclusion, we find that ECINN outperforms competing methods on both quality and speed and comply with our desiderata by realistically changing the predicted and the perceived class while maintaining class-independent features such as font-weight and tilt.

4.3 CelebA-HQ.

To evaluate ECINN on a more diverse and complex dataset, we extend our experiments to the CelebA-HQ dataset. We train IB-INNs to predict various binary labels, where each class is represented by at least 45% of the dataset.

In Figure 6, we show counterfactual examples for the smile versus frown label. The first five columns depict how ECINN turns frowning people into smiling ones, while the last five columns make smiling people frown. First, we observe that class irrelevant features such as hair, skin color, and backgrounds remain perceptually unchanged as desired. Second, we notice that some counterfactual examples in the last row look unrealistic. In particular, it seems hard for the method to open and close mouths. In some cases, we also observe small artifacts like the ones in the left-most pixels of the second column. Based on our MNIST experiments, which did not suffer from computational limitations, we believe that scaling from the roughly 40 million parameters used to around 200 million (as is common with previous work [19]) can remove the artifacts and generate higher quality counterfactual examples. Furthermore, the low-resolution version of CelebA-HQ that we use due to limited resources is arguably harder to synthesize than higher resolutions. For further verification of our findings, we include plots for models trained on other labels in Section E of the appendix.

5 Conclusion

We introduce ECINN as an efficient method for computing counterfactual examples. Our method is derived from theoretical and practical properties of a particular type of classifiers, namely conditional INNs. While being three orders of magnitude faster than competing methods, ECINN requires only one forward and one inverse pass, it generates a unique solution, and it requires no numerical optimization. In compliance with our desiderata, ECINN generates counterfactual explanations that i) change only class-dependent features, ii) are realistic, and iii) can represent both tipping-point and convincing counterfactuals.

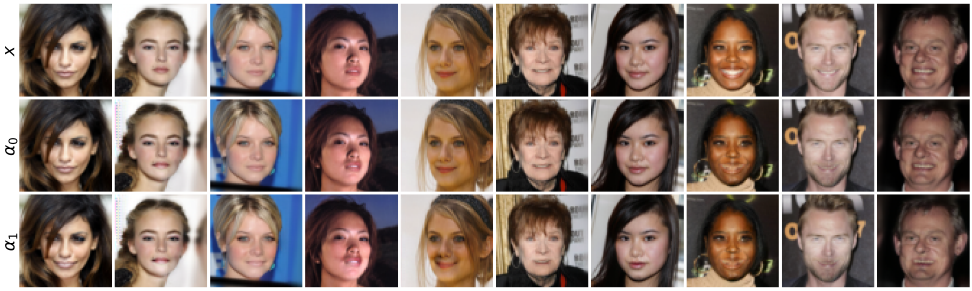


Figure 6: Counterfactual examples for frowning and smiling faces. First five columns have target $q =$ smile and last five columns $q =$ frown. $p(q|x) > 1 - 10^{-4}$ for all samples.

References

[1] Arjun Akula, Shuai Wang, and Song-Chun Zhu. CoCoX: Generating Conceptual and Counterfactual Explanations via Fault-Lines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[2] Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. *NeurIPS*, 2020.

[3] 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*, 2015.

[4] Jens Behrmann, Will Grathwohl, Ricky T.Q. Chen, David Duvenaud, and Jörn Henrik Jacobsen. Invertible residual networks. In *ICML*, 2019.

[5] Yoshua Bengio and Yann LeCun. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

[6] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *arXiv preprint arXiv:2102.13076*, 2021.

[7] Stuart K Card, George G Robertson, and Jock D Mackinlay. The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pages 181–186, 1991.

[8] Furui Cheng, Yao Ming, and Huamin Qu. DECE: decision explorer with counterfactual explanations for machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1438–1447, 2021.

[9] 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 *NeurIPS*, 2018.

[10] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-Linear Independent Components Estimation. In *ICLR (Workshop)*, 2015.

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2019.

[12] Oscar Gomez, Steffen Holter, Jun Yuan, and Enrico Bertini. ViCE: Visual Counterfactual Explanations for Machine Learning Models. *International Conference on Intelligent User Interfaces, Proceedings IUI*, pages 531–535, 2020.

[13] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual Visual Explanations. In *ICML*, 2019.

[14] Pavel Izmailov, Polina Kirichenko, Marc Finzi, and Andrew Gordon Wilson. Semi-supervised learning with normalizing flows. In *ICML*, 2020.

[15] Alon Jacovi, Swabha Swayamdipta, Shauli Ravfogel, Yanai Elazar, Yejin Choi, and Yoav Goldberg. Contrastive Explanations for Model Interpretability. *arXiv preprint arXiv:2103.01378*, 2021.

[16] Sin-Han Kang, Honggyu Jung, Dong-Ok Won, and Seong-Whan Lee. Counterfactual explanation based on gradual construction for deep networks. *arXiv preprint arXiv:2008.01897*, 2020.

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *ICLR*, 2018.

[18] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*, 2018.

[20] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

[21] Radek Mackowiak, Lynton Ardizzone, Ullrich Köthe, and Carsten Rother. Generative classifiers as a basis for trustworthy computer vision. *arXiv preprint arXiv:2007.15036*, 2020.

[22] Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. In *CausalML: Machine Learning and Causal Inference for Improved Decision Making Workshop, NeurIPS 2019*, December 2019.

[23] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015.

[24] Tan M. Nguyen, Animesh Garg, Richard G. Baraniuk, and Anima Anandkumar. InfoCNF: Efficient conditional continuous normalizing flow using adaptive solvers. *arXiv preprint arXiv:1912.03978*, 2019.

[25] Jingjing Pan, Yash Goyal, and Stefan Lee. Question-conditioned counterfactual image generation for VQA. *arXiv preprint arXiv:1911.06352*, 2019.

[26] Judea Pearl. Causes of effects and effects of causes. *Sociological Methods & Research*, 44(1):149–164, 2015.

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1 (8):9, 2019.

[28] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. CERTIFAI: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence models. *CoRR*, 2019. URL http://arxiv.org/abs/1905.07857.

[29] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014.

[30] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23 (5):828–841, 2019.

[31] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

[32] Tom Vermeire and David Martens. Explainable image classification with evidence counterfactual. *arXiv preprint arXiv:2004.07511*, 2020.

[33] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31:841, 2017.

[34] Pei Wang and Nuno Vasconcelos. SCOUT: Self-aware Discriminant Counterfactual Explanations. In *CVPR*, 2020.

[35] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S. Weld. Polyjuice: Automated, General-purpose Counterfactual Generation. *arXiv preprint arXiv:2101.00288*, 2021.